In [2]:
```
#!source dl_workshop/bin/activate
```

In [3]:
```
import tensorflow
print('Tensorflow version: ', tensorflow.__version__)
```

```
Tensorflow version:  2.17.0
```

In [4]:
```
import seaborn as sns
```

In [5]:
```
penguins = sns.load_dataset('penguins')
```

In [6]:
```
penguins.head()
```

Out[6]:

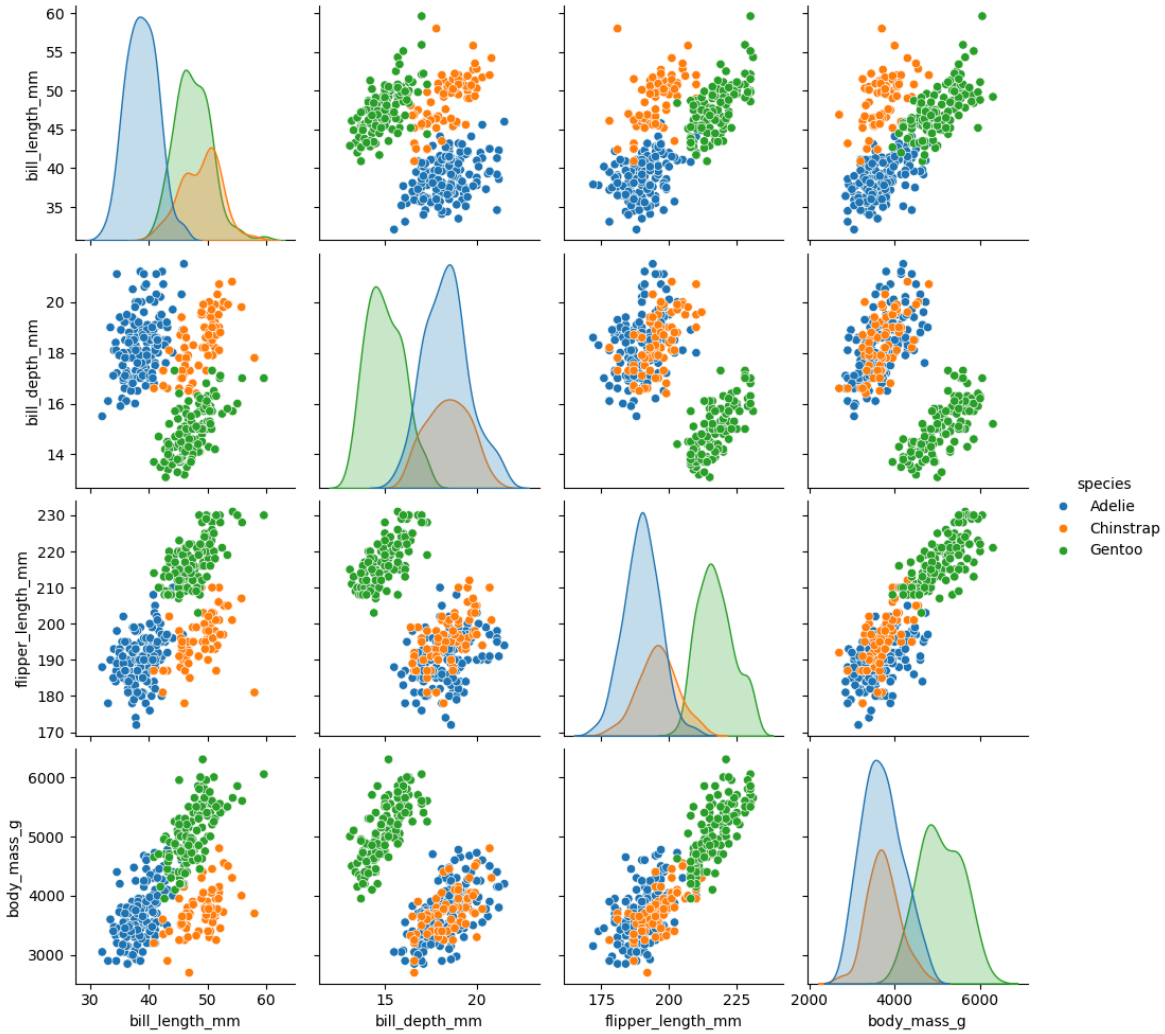
	species	island	bill_length_mm	bill_depth_mm	flipper_length_mm	body_m
0	Adelie	Torgersen	39.1	18.7	181.0	3
1	Adelie	Torgersen	39.5	17.4	186.0	3
2	Adelie	Torgersen	40.3	18.0	195.0	3
3	Adelie	Torgersen	NaN	NaN	NaN	
4	Adelie	Torgersen	36.7	19.3	193.0	3

In [7]:
```
penguins.shape
```

Out[7]:
```
(344, 7)
```

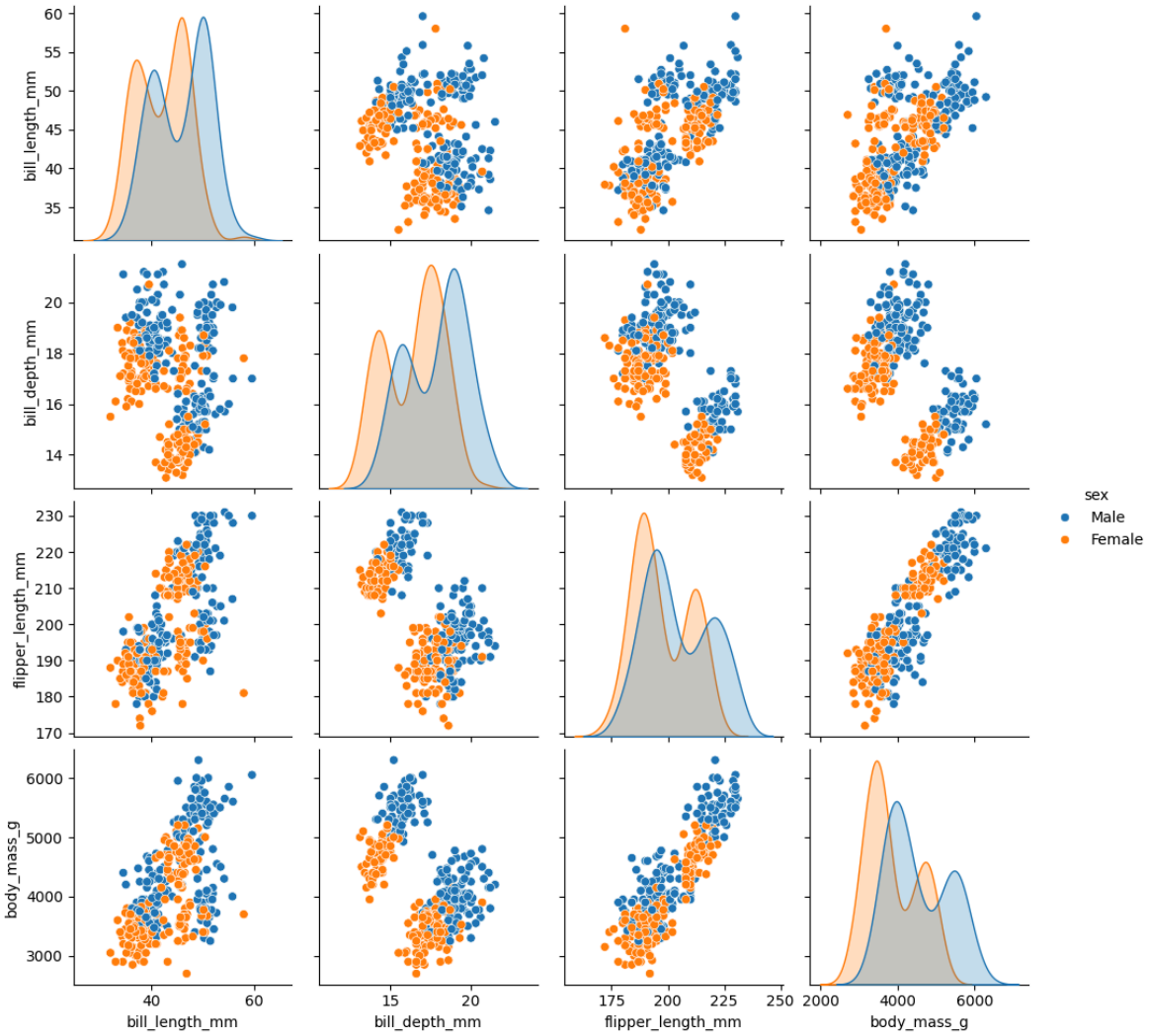
In [8]:
```
sns.pairplot(penguins, hue ="species")
```

Out[8]:
```
<seaborn.axisgrid.PairGrid at 0x166b9bed0>
```



In [9]:
```
sns.pairplot(penguins, hue ="sex")
```

Out[9]: <seaborn.axisgrid.PairGrid at 0x167a866d0>



In [10]:
```
penguins_filtered =penguins.drop(columns=['island', 'sex'])
```

In [11]:
```
penguins_filtered = penguins_filtered.dropna()
```

In [12]:
```
features = penguins_filtered.drop(columns=['species'])
```

In [13]:
```
import pandas as pd

target = pd.get_dummies(penguins_filtered['species'])
target.head() # print out the top 5 to see what it looks like.
```

Out[13]:

	Adelie	Chinstrap	Gentoo
0	True	False	False
1	True	False	False
2	True	False	False
4	True	False	False
5	True	False	False

In [14]:
```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(features, target, tes
```

In [15]:
```
from tensorflow import keras
```

In [16]:
```
from numpy.random import seed
seed(1)
keras.utils.set_random_seed(2)
```

In [17]:
```
inputs = keras.Input(shape=(X_train.shape[1],))
```

In [18]:
```
hidden_layer = keras.layers.Dense(10, activation="relu")(inputs)
```

In [19]:
```
output_layer = keras.layers.Dense(3, activation="softmax")(hidden_layer)
```

In [20]:
```
model = keras.Model(inputs=inputs, outputs=output_layer)
model.summary()
```

Model: "functional"

Layer (type)	Output Shape	P
input_layer (InputLayer)	(None, 4)	
dense (Dense)	(None, 10)	
dense_1 (Dense)	(None, 3)	

Total params: 83 (332.00 B)
Trainable params: 83 (332.00 B)
Non-trainable params: 0 (0.00 B)

In [21]:
```
model.compile(optimizer='adam', loss=keras.losses.CategoricalCrossentropy
```

In [22]:
```
history = model.fit(X_train, y_train, epochs=100)
```

```
Epoch 1/100
9/9 ━━━━━━━━━━━━━━━━━━━━ 0s 541us/step - loss: 734.7374
Epoch 2/100
9/9 ━━━━━━━━━━━━━━━━━━━━ 0s 332us/step - loss: 616.7337
Epoch 3/100
9/9 ━━━━━━━━━━━━━━━━━━━━ 0s 306us/step - loss: 497.1732
Epoch 4/100
9/9 ━━━━━━━━━━━━━━━━━━━━ 0s 317us/step - loss: 375.0363
Epoch 5/100
9/9 ━━━━━━━━━━━━━━━━━━━━ 0s 317us/step - loss: 254.4783
Epoch 6/100
9/9 ━━━━━━━━━━━━━━━━━━━━ 0s 312us/step - loss: 164.8771
Epoch 7/100
9/9 ━━━━━━━━━━━━━━━━━━━━ 0s 309us/step - loss: 83.2320
Epoch 8/100
9/9 ━━━━━━━━━━━━━━━━━━━━ 0s 310us/step - loss: 24.4818
Epoch 9/100
9/9 ━━━━━━━━━━━━━━━━━━━━ 0s 311us/step - loss: 18.2118
Epoch 10/100
9/9 ━━━━━━━━━━━━━━━━━━━━ 0s 322us/step - loss: 23.7708
Epoch 11/100
9/9 ━━━━━━━━━━━━━━━━━━━━ 0s 279us/step - loss: 15.8151
Epoch 12/100
9/9 ━━━━━━━━━━━━━━━━━━━━ 0s 277us/step - loss: 15.6675
Epoch 13/100
9/9 ━━━━━━━━━━━━━━━━━━━━ 0s 280us/step - loss: 15.5323
Epoch 14/100
9/9 ━━━━━━━━━━━━━━━━━━━━ 0s 272us/step - loss: 14.9596
Epoch 15/100
9/9 ━━━━━━━━━━━━━━━━━━━━ 0s 271us/step - loss: 15.2996
Epoch 16/100
9/9 ━━━━━━━━━━━━━━━━━━━━ 0s 282us/step - loss: 14.6142
Epoch 17/100
9/9 ━━━━━━━━━━━━━━━━━━━━ 0s 296us/step - loss: 14.7340
Epoch 18/100
9/9 ━━━━━━━━━━━━━━━━━━━━ 0s 278us/step - loss: 14.5965
Epoch 19/100
9/9 ━━━━━━━━━━━━━━━━━━━━ 0s 283us/step - loss: 14.5240
Epoch 20/100
9/9 ━━━━━━━━━━━━━━━━━━━━ 0s 281us/step - loss: 14.2889
Epoch 21/100
9/9 ━━━━━━━━━━━━━━━━━━━━ 0s 277us/step - loss: 14.1589
Epoch 22/100
9/9 ━━━━━━━━━━━━━━━━━━━━ 0s 286us/step - loss: 13.9980
Epoch 23/100
9/9 ━━━━━━━━━━━━━━━━━━━━ 0s 270us/step - loss: 13.8486
Epoch 24/100
9/9 ━━━━━━━━━━━━━━━━━━━━ 0s 285us/step - loss: 13.6963
Epoch 25/100
9/9 ━━━━━━━━━━━━━━━━━━━━ 0s 296us/step - loss: 13.5231
Epoch 26/100
9/9 ━━━━━━━━━━━━━━━━━━━━ 0s 372us/step - loss: 13.3674
Epoch 27/100
9/9 ━━━━━━━━━━━━━━━━━━━━ 0s 319us/step - loss: 13.1892
Epoch 28/100
9/9 ━━━━━━━━━━━━━━━━━━━━ 0s 290us/step - loss: 13.0328
Epoch 29/100
9/9 ━━━━━━━━━━━━━━━━━━━━ 0s 244us/step - loss: 12.8535
Epoch 30/100
9/9 ━━━━━━━━━━━━━━━━━━━━ 0s 283us/step - loss: 12.6921
```

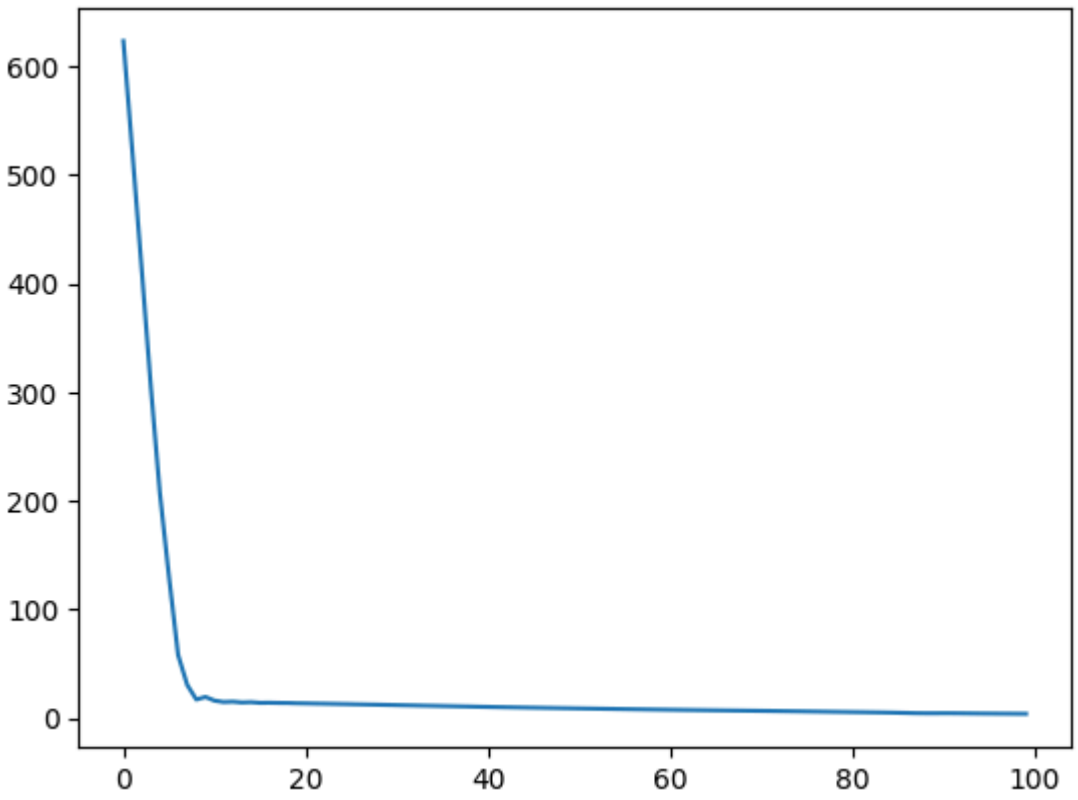
```
Epoch 31/100
9/9 ━━━━━━━━━━━━━━━━━━━━ 0s 294us/step - loss: 12.5142
Epoch 32/100
9/9 ━━━━━━━━━━━━━━━━━━━━ 0s 298us/step - loss: 12.3486
Epoch 33/100
9/9 ━━━━━━━━━━━━━━━━━━━━ 0s 318us/step - loss: 12.1747
Epoch 34/100
9/9 ━━━━━━━━━━━━━━━━━━━━ 0s 297us/step - loss: 12.0072
Epoch 35/100
9/9 ━━━━━━━━━━━━━━━━━━━━ 0s 307us/step - loss: 11.8382
Epoch 36/100
9/9 ━━━━━━━━━━━━━━━━━━━━ 0s 390us/step - loss: 11.6720
Epoch 37/100
9/9 ━━━━━━━━━━━━━━━━━━━━ 0s 314us/step - loss: 11.5079
Epoch 38/100
9/9 ━━━━━━━━━━━━━━━━━━━━ 0s 322us/step - loss: 11.3455
Epoch 39/100
9/9 ━━━━━━━━━━━━━━━━━━━━ 0s 294us/step - loss: 11.1860
Epoch 40/100
9/9 ━━━━━━━━━━━━━━━━━━━━ 0s 301us/step - loss: 11.0294
Epoch 41/100
9/9 ━━━━━━━━━━━━━━━━━━━━ 0s 290us/step - loss: 10.8766
Epoch 42/100
9/9 ━━━━━━━━━━━━━━━━━━━━ 0s 292us/step - loss: 10.7283
Epoch 43/100
9/9 ━━━━━━━━━━━━━━━━━━━━ 0s 297us/step - loss: 10.5848
Epoch 44/100
9/9 ━━━━━━━━━━━━━━━━━━━━ 0s 298us/step - loss: 10.4455
Epoch 45/100
9/9 ━━━━━━━━━━━━━━━━━━━━ 0s 319us/step - loss: 10.3095
Epoch 46/100
9/9 ━━━━━━━━━━━━━━━━━━━━ 0s 291us/step - loss: 10.1754
Epoch 47/100
9/9 ━━━━━━━━━━━━━━━━━━━━ 0s 294us/step - loss: 10.0429
Epoch 48/100
9/9 ━━━━━━━━━━━━━━━━━━━━ 0s 302us/step - loss: 9.9120
Epoch 49/100
9/9 ━━━━━━━━━━━━━━━━━━━━ 0s 282us/step - loss: 9.7827
Epoch 50/100
9/9 ━━━━━━━━━━━━━━━━━━━━ 0s 361us/step - loss: 9.6543
Epoch 51/100
9/9 ━━━━━━━━━━━━━━━━━━━━ 0s 318us/step - loss: 9.5262
Epoch 52/100
9/9 ━━━━━━━━━━━━━━━━━━━━ 0s 308us/step - loss: 9.3973
Epoch 53/100
9/9 ━━━━━━━━━━━━━━━━━━━━ 0s 326us/step - loss: 9.2678
Epoch 54/100
9/9 ━━━━━━━━━━━━━━━━━━━━ 0s 321us/step - loss: 9.1381
Epoch 55/100
9/9 ━━━━━━━━━━━━━━━━━━━━ 0s 290us/step - loss: 9.0094
Epoch 56/100
9/9 ━━━━━━━━━━━━━━━━━━━━ 0s 279us/step - loss: 8.8826
Epoch 57/100
9/9 ━━━━━━━━━━━━━━━━━━━━ 0s 306us/step - loss: 8.7584
Epoch 58/100
9/9 ━━━━━━━━━━━━━━━━━━━━ 0s 290us/step - loss: 8.6374
Epoch 59/100
9/9 ━━━━━━━━━━━━━━━━━━━━ 0s 297us/step - loss: 8.5198
Epoch 60/100
9/9 ━━━━━━━━━━━━━━━━━━━━ 0s 289us/step - loss: 8.4062
```

```
Epoch 61/100
9/9 ━━━━━━━━━━━━━━━━━━━━ 0s 310us/step - loss: 8.2966
Epoch 62/100
9/9 ━━━━━━━━━━━━━━━━━━━━ 0s 289us/step - loss: 8.1911
Epoch 63/100
9/9 ━━━━━━━━━━━━━━━━━━━━ 0s 289us/step - loss: 8.0899
Epoch 64/100
9/9 ━━━━━━━━━━━━━━━━━━━━ 0s 291us/step - loss: 7.9932
Epoch 65/100
9/9 ━━━━━━━━━━━━━━━━━━━━ 0s 303us/step - loss: 7.9011
Epoch 66/100
9/9 ━━━━━━━━━━━━━━━━━━━━ 0s 290us/step - loss: 7.8134
Epoch 67/100
9/9 ━━━━━━━━━━━━━━━━━━━━ 0s 299us/step - loss: 7.7287
Epoch 68/100
9/9 ━━━━━━━━━━━━━━━━━━━━ 0s 320us/step - loss: 7.6463
Epoch 69/100
9/9 ━━━━━━━━━━━━━━━━━━━━ 0s 295us/step - loss: 7.5640
Epoch 70/100
9/9 ━━━━━━━━━━━━━━━━━━━━ 0s 276us/step - loss: 7.4804
Epoch 71/100
9/9 ━━━━━━━━━━━━━━━━━━━━ 0s 313us/step - loss: 7.3942
Epoch 72/100
9/9 ━━━━━━━━━━━━━━━━━━━━ 0s 307us/step - loss: 7.3043
Epoch 73/100
9/9 ━━━━━━━━━━━━━━━━━━━━ 0s 280us/step - loss: 7.2102
Epoch 74/100
9/9 ━━━━━━━━━━━━━━━━━━━━ 0s 295us/step - loss: 7.1116
Epoch 75/100
9/9 ━━━━━━━━━━━━━━━━━━━━ 0s 297us/step - loss: 7.0081
Epoch 76/100
9/9 ━━━━━━━━━━━━━━━━━━━━ 0s 293us/step - loss: 6.9002
Epoch 77/100
9/9 ━━━━━━━━━━━━━━━━━━━━ 0s 302us/step - loss: 6.7874
Epoch 78/100
9/9 ━━━━━━━━━━━━━━━━━━━━ 0s 369us/step - loss: 6.6671
Epoch 79/100
9/9 ━━━━━━━━━━━━━━━━━━━━ 0s 301us/step - loss: 6.5354
Epoch 80/100
9/9 ━━━━━━━━━━━━━━━━━━━━ 0s 303us/step - loss: 6.3951
Epoch 81/100
9/9 ━━━━━━━━━━━━━━━━━━━━ 0s 297us/step - loss: 6.2552
Epoch 82/100
9/9 ━━━━━━━━━━━━━━━━━━━━ 0s 315us/step - loss: 6.1217
Epoch 83/100
9/9 ━━━━━━━━━━━━━━━━━━━━ 0s 317us/step - loss: 5.9905
Epoch 84/100
9/9 ━━━━━━━━━━━━━━━━━━━━ 0s 304us/step - loss: 5.8509
Epoch 85/100
9/9 ━━━━━━━━━━━━━━━━━━━━ 0s 296us/step - loss: 5.6888
Epoch 86/100
9/9 ━━━━━━━━━━━━━━━━━━━━ 0s 321us/step - loss: 5.4871
Epoch 87/100
9/9 ━━━━━━━━━━━━━━━━━━━━ 0s 311us/step - loss: 5.2390
Epoch 88/100
9/9 ━━━━━━━━━━━━━━━━━━━━ 0s 373us/step - loss: 4.9852
Epoch 89/100
9/9 ━━━━━━━━━━━━━━━━━━━━ 0s 317us/step - loss: 4.8175
Epoch 90/100
9/9 ━━━━━━━━━━━━━━━━━━━━ 0s 306us/step - loss: 4.7881
```

```
Epoch 91/100
9/9 ━━━━━━━━━━━━━━━━━━━━ 0s 301us/step - loss: 4.8190
Epoch 92/100
9/9 ━━━━━━━━━━━━━━━━━━━━ 0s 288us/step - loss: 4.8231
Epoch 93/100
9/9 ━━━━━━━━━━━━━━━━━━━━ 0s 293us/step - loss: 4.7515
Epoch 94/100
9/9 ━━━━━━━━━━━━━━━━━━━━ 0s 299us/step - loss: 4.6527
Epoch 95/100
9/9 ━━━━━━━━━━━━━━━━━━━━ 0s 296us/step - loss: 4.5632
Epoch 96/100
9/9 ━━━━━━━━━━━━━━━━━━━━ 0s 302us/step - loss: 4.4825
Epoch 97/100
9/9 ━━━━━━━━━━━━━━━━━━━━ 0s 291us/step - loss: 4.4120
Epoch 98/100
9/9 ━━━━━━━━━━━━━━━━━━━━ 0s 320us/step - loss: 4.3448
Epoch 99/100
9/9 ━━━━━━━━━━━━━━━━━━━━ 0s 302us/step - loss: 4.2768
Epoch 100/100
9/9 ━━━━━━━━━━━━━━━━━━━━ 0s 289us/step - loss: 4.2095
```

In [23]:
```
sns.lineplot(x=history.epoch, y=history.history['loss'])
```

Out[23]: <Axes: >



In [24]:
```
y_pred = model.predict(X_test)
prediction = pd.DataFrame(y_pred, columns=target.columns)
prediction
```

```
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step
```

Out[24]:

	Adelie	Chinstrap	Gentoo
0	0.999997	3.051823e-06	3.197025e-07
1	0.988953	1.051394e-02	5.330010e-04
2	0.999905	9.425520e-05	1.243060e-06
3	0.997220	1.081525e-03	1.698676e-03
4	0.857294	6.665846e-02	7.604740e-02
...	...	...	...
64	0.013076	9.671507e-01	1.977344e-02
65	0.204562	7.900265e-01	5.411607e-03
66	0.107249	5.037869e-01	3.889638e-01
67	0.999680	5.593414e-05	2.640279e-04
68	0.999999	9.286309e-08	8.178871e-07

69 rows × 3 columns

In [25]:
```
predicted_species = prediction.idxmax(axis="columns")
predicted_species
```

Out[25]:
```
0        Adelie
1        Adelie
2        Adelie
3        Adelie
4        Adelie
        ...
64    Chinstrap
65    Chinstrap
66    Chinstrap
67       Adelie
68       Adelie
Length: 69, dtype: object
```

In [26]:
```
from sklearn.metrics import confusion_matrix

true_species = y_test.idxmax(axis="columns")

matrix = confusion_matrix(true_species, predicted_species)
print(matrix)
```

```
[[16 14  0]
 [ 3  4  7]
 [25  0  0]]
```

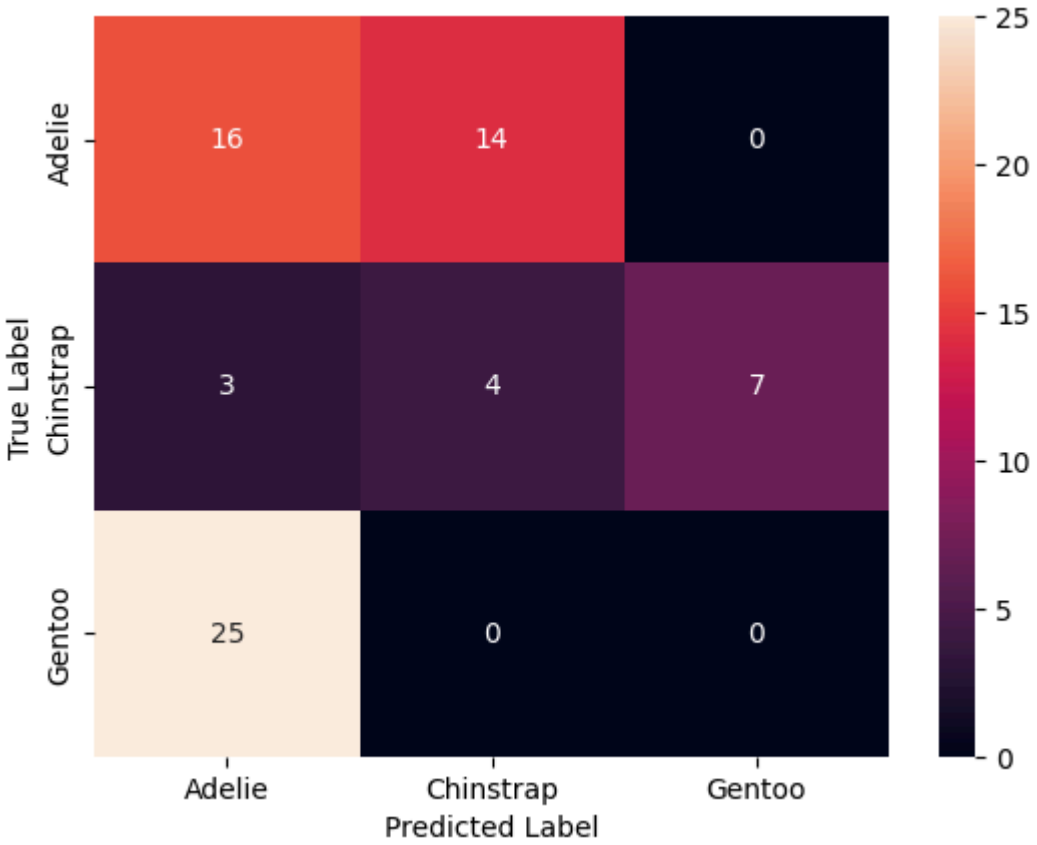
In [27]:
```
# Convert to a pandas dataframe
confusion_df = pd.DataFrame(matrix, index=y_test.columns.values, columns=

# Set the names of the x and y axis, this helps with the readability of t
confusion_df.index.name = 'True Label'
confusion_df.columns.name = 'Predicted Label'
confusion_df.head()
```

Out[27]:

Predicted Label	Adelie	Chinstrap	Gentoo
True Label			
Adelie	16	14	0
Chinstrap	3	4	7
Gentoo	25	0	0

In [28]:
```
sns.heatmap(confusion_df, annot=True)
```

Out[28]:
```
<Axes: xlabel='Predicted Label', ylabel='True Label'>
```



In [29]:
```
model.save('my_first_model.keras')
```

In [30]:
```
pretrained_model = keras.models.load_model('my_first_model.keras')
```

In [31]:
```
# use the pretrained model here
y_pretrained_pred = pretrained_model.predict(X_test)
pretrained_prediction = pd.DataFrame(y_pretrained_pred, columns=target.co

# idxmax will select the column for each row with the highest value
pretrained_predicted_species = pretrained_prediction.idxmax(axis="columns
print(pretrained_predicted_species)
```

```
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step
0        Adelie
1        Adelie
2        Adelie
3        Adelie
4        Adelie
        ...
64    Chinstrap
65    Chinstrap
66    Chinstrap
67       Adelie
68       Adelie
Length: 69, dtype: object
```

In [32]:
```
data = pd.read_csv("https://zenodo.org/record/5071376/files/weather_predi
```

```
---------------------------------------------------------------------------
SSLCertVerificationError                  Traceback (most recent call last)
File /Library/Frameworks/Python.framework/Versions/3.11/lib/python3.11/urllib/request.py:1348, in AbstractHTTPHandler.do_open(self, http_class, req, **http_conn_args)
   1347 try:
-> 1348     h.request(req.get_method(), req.selector, req.data, headers,
   1349               encode_chunked=req.has_header('Transfer-encoding'))
   1350 except OSError as err: # timeout error

File /Library/Frameworks/Python.framework/Versions/3.11/lib/python3.11/http/client.py:1286, in HTTPConnection.request(self, method, url, body, headers, encode_chunked)
   1285 """Send a complete request to the server."""
-> 1286 self._send_request(method, url, body, headers, encode_chunked)

File /Library/Frameworks/Python.framework/Versions/3.11/lib/python3.11/http/client.py:1332, in HTTPConnection._send_request(self, method, url, body, headers, encode_chunked)
   1331     body = _encode(body, 'body')
-> 1332 self.endheaders(body, encode_chunked=encode_chunked)

File /Library/Frameworks/Python.framework/Versions/3.11/lib/python3.11/http/client.py:1281, in HTTPConnection.endheaders(self, message_body, encode_chunked)
   1280     raise CannotSendHeader()
-> 1281 self._send_output(message_body, encode_chunked=encode_chunked)

File /Library/Frameworks/Python.framework/Versions/3.11/lib/python3.11/http/client.py:1041, in HTTPConnection._send_output(self, message_body, encode_chunked)
   1040 del self._buffer[:]
-> 1041 self.send(msg)
   1043 if message_body is not None:
   1044
   1045     # create a consistent interface to message_body

File /Library/Frameworks/Python.framework/Versions/3.11/lib/python3.11/http/client.py:979, in HTTPConnection.send(self, data)
    978 if self.auto_open:
--> 979     self.connect()
    980 else:

File /Library/Frameworks/Python.framework/Versions/3.11/lib/python3.11/http/client.py:1458, in HTTPSConnection.connect(self)
   1456     server_hostname = self.host
-> 1458 self.sock = self._context.wrap_socket(self.sock,
   1459                                       server_hostname=server_hostname)

File /Library/Frameworks/Python.framework/Versions/3.11/lib/python3.11/ssl.py:517, in SSLContext.wrap_socket(self, sock, server_side, do_handshake_on_connect, suppress_ragged_eofs, server_hostname, session)
    511 def wrap_socket(self, sock, server_side=False,
    512                 do_handshake_on_connect=True,
    513                 suppress_ragged_eofs=True,
    514                 server_hostname=None, session=None):
    515     # SSLSocket class handles server_hostname encoding before it c
```

```
alls
    516     # ctx._wrap_socket()
--> 517     return self.sslsocket_class._create(
    518         sock=sock,
    519         server_side=server_side,
    520         do_handshake_on_connect=do_handshake_on_connect,
    521         suppress_ragged_eofs=suppress_ragged_eofs,
    522         server_hostname=server_hostname,
    523         context=self,
    524         session=session
    525     )

File /Library/Frameworks/Python.framework/Versions/3.11/lib/python3.11/ssl.py:1075, in SSLSocket._create(cls, sock, server_side, do_handshake_on_connect, suppress_ragged_eofs, server_hostname, context, session)
   1074             raise ValueError("do_handshake_on_connect should not be specified for non-blocking sockets")
-> 1075         self.do_handshake()
   1076 except (OSError, ValueError):

File /Library/Frameworks/Python.framework/Versions/3.11/lib/python3.11/ssl.py:1346, in SSLSocket.do_handshake(self, block)
   1345         self.settimeout(None)
-> 1346     self._sslobj.do_handshake()
   1347 finally:

SSLCertVerificationError: [SSL: CERTIFICATE_VERIFY_FAILED] certificate verify failed: unable to get local issuer certificate (_ssl.c:1002)

During handling of the above exception, another exception occurred:

URLError                                  Traceback (most recent call last)
Cell In[32], line 1
----> 1 data = pd.read_csv("https://zenodo.org/record/5071376/files/weather_prediction_dataset_light.csv?download=1")

File ~/Documents/dl_workshop/lib/python3.11/site-packages/pandas/io/parsers/readers.py:1026, in read_csv(filepath_or_buffer, sep, delimiter, header, names, index_col, usecols, dtype, engine, converters, true_values, false_values, skipinitialspace, skiprows, skipfooter, nrows, na_values, keep_default_na, na_filter, verbose, skip_blank_lines, parse_dates, infer_datetime_format, keep_date_col, date_parser, date_format, dayfirst, cache_dates, iterator, chunksize, compression, thousands, decimal, lineterminator, quotechar, quoting, doublequote, escapechar, comment, encoding, encoding_errors, dialect, on_bad_lines, delim_whitespace, low_memory, memory_map, float_precision, storage_options, dtype_backend)
   1013 kwds_defaults = _refine_defaults_read(
   1014     dialect,
   1015     delimiter,
   (...)
   1022     dtype_backend=dtype_backend,
   1023 )
   1024 kwds.update(kwds_defaults)
-> 1026 return _read(filepath_or_buffer, kwds)

File ~/Documents/dl_workshop/lib/python3.11/site-packages/pandas/io/parsers/readers.py:620, in _read(filepath_or_buffer, kwds)
    617 _validate_names(kwds.get("names", None))
    619 # Create the parser.
```

```
--> 620 parser = TextFileReader(filepath_or_buffer, **kwds)
    622 if chunksize or iterator:
    623     return parser

File ~/Documents/dl_workshop/lib/python3.11/site-packages/pandas/io/parsers/readers.py:1620, in TextFileReader.__init__(self, f, engine, **kwds)
   1617     self.options["has_index_names"] = kwds["has_index_names"]
   1619 self.handles: IOHandles | None = None
-> 1620 self._engine = self._make_engine(f, self.engine)

File ~/Documents/dl_workshop/lib/python3.11/site-packages/pandas/io/parsers/readers.py:1880, in TextFileReader._make_engine(self, f, engine)
   1878     if "b" not in mode:
   1879         mode += "b"
-> 1880 self.handles = get_handle(
   1881     f,
   1882     mode,
   1883     encoding=self.options.get("encoding", None),
   1884     compression=self.options.get("compression", None),
   1885     memory_map=self.options.get("memory_map", False),
   1886     is_text=is_text,
   1887     errors=self.options.get("encoding_errors", "strict"),
   1888     storage_options=self.options.get("storage_options", None),
   1889 )
   1890 assert self.handles is not None
   1891 f = self.handles.handle

File ~/Documents/dl_workshop/lib/python3.11/site-packages/pandas/io/common.py:728, in get_handle(path_or_buf, mode, encoding, compression, memory_map, is_text, errors, storage_options)
    725     codecs.lookup_error(errors)
    727 # open URLs
--> 728 ioargs = _get_filepath_or_buffer(
    729     path_or_buf,
    730     encoding=encoding,
    731     compression=compression,
    732     mode=mode,
    733     storage_options=storage_options,
    734 )
    736 handle = ioargs.filepath_or_buffer
    737 handles: list[BaseBuffer]

File ~/Documents/dl_workshop/lib/python3.11/site-packages/pandas/io/common.py:384, in _get_filepath_or_buffer(filepath_or_buffer, encoding, compression, mode, storage_options)
    382 # assuming storage_options is to be interpreted as headers
    383 req_info = urllib.request.Request(filepath_or_buffer, headers=storage_options)
--> 384 with urlopen(req_info) as req:
    385     content_encoding = req.headers.get("Content-Encoding", None)
    386     if content_encoding == "gzip":
    387         # Override compression based on Content-Encoding header

File ~/Documents/dl_workshop/lib/python3.11/site-packages/pandas/io/common.py:289, in urlopen(*args, **kwargs)
    283 """
    284 Lazy-import wrapper for stdlib urlopen, as that imports a big chunk of
    285 the stdlib.
    286 """
```

```
    287 import urllib.request
--> 289 return urllib.request.urlopen(*args, **kwargs)

File /Library/Frameworks/Python.framework/Versions/3.11/lib/python3.11/url
lib/request.py:216, in urlopen(url, data, timeout, cafile, capath, cadefau
lt, context)
    214 else:
    215     opener = _opener
--> 216 return opener.open(url, data, timeout)

File /Library/Frameworks/Python.framework/Versions/3.11/lib/python3.11/url
lib/request.py:519, in OpenerDirector.open(self, fullurl, data, timeout)
    516     req = meth(req)
    518 sys.audit('urllib.Request', req.full_url, req.data, req.headers, r
eq.get_method())
--> 519 response = self._open(req, data)
    521 # post-process response
    522 meth_name = protocol+"_response"

File /Library/Frameworks/Python.framework/Versions/3.11/lib/python3.11/url
lib/request.py:536, in OpenerDirector._open(self, req, data)
    533     return result
    535 protocol = req.type
--> 536 result = self._call_chain(self.handle_open, protocol, protocol +
    537                           '_open', req)
    538 if result:
    539     return result

File /Library/Frameworks/Python.framework/Versions/3.11/lib/python3.11/url
lib/request.py:496, in OpenerDirector._call_chain(self, chain, kind, meth_
name, *args)
    494 for handler in handlers:
    495     func = getattr(handler, meth_name)
--> 496     result = func(*args)
    497     if result is not None:
    498         return result

File /Library/Frameworks/Python.framework/Versions/3.11/lib/python3.11/url
lib/request.py:1391, in HTTPSHandler.https_open(self, req)
   1390 def https_open(self, req):
-> 1391     return self.do_open(http.client.HTTPSConnection, req,
   1392         context=self._context, check_hostname=self._check_hostnam
e)

File /Library/Frameworks/Python.framework/Versions/3.11/lib/python3.11/url
lib/request.py:1351, in AbstractHTTPHandler.do_open(self, http_class, req,
**http_conn_args)
   1348         h.request(req.get_method(), req.selector, req.data, header
s,
   1349                   encode_chunked=req.has_header('Transfer-encodin
g'))
   1350     except OSError as err: # timeout error
-> 1351         raise URLError(err)
   1352     r = h.getresponse()
   1353 except:

URLError: <urlopen error [SSL: CERTIFICATE_VERIFY_FAILED] certificate veri
fy failed: unable to get local issuer certificate (_ssl.c:1002)>
```

In [33]:
```
import pandas as pd

filename_data = "weather_prediction_dataset_light.csv"
data = pd.read_csv(filename_data)
data.head()
```

Out[33]:

	DATE	MONTH	BASEL_cloud_cover	BASEL_humidity	BASEL_pressure	BAS
0	20000101	1	8	0.89	1.0286	
1	20000102	1	8	0.87	1.0318	
2	20000103	1	5	0.81	1.0314	
3	20000104	1	7	0.79	1.0262	
4	20000105	1	5	0.90	1.0246	

5 rows × 91 columns

In [34]:
```
data.columns
```

Out[34]:
```
Index(['DATE', 'MONTH', 'BASEL_cloud_cover', 'BASEL_humidity',
       'BASEL_pressure', 'BASEL_global_radiation', 'BASEL_precipitatio
n',
       'BASEL_sunshine', 'BASEL_temp_mean', 'BASEL_temp_min', 'BASEL_tem
p_max',
       'DE_BILT_cloud_cover', 'DE_BILT_humidity', 'DE_BILT_pressure',
       'DE_BILT_global_radiation', 'DE_BILT_precipitation', 'DE_BILT_sun
shine',
       'DE_BILT_temp_mean', 'DE_BILT_temp_min', 'DE_BILT_temp_max',
       'DRESDEN_cloud_cover', 'DRESDEN_humidity', 'DRESDEN_global_radiat
ion',
       'DRESDEN_precipitation', 'DRESDEN_sunshine', 'DRESDEN_temp_mean',
       'DRESDEN_temp_min', 'DRESDEN_temp_max', 'DUSSELDORF_cloud_cover',
       'DUSSELDORF_humidity', 'DUSSELDORF_pressure',
       'DUSSELDORF_global_radiation', 'DUSSELDORF_precipitation',
       'DUSSELDORF_sunshine', 'DUSSELDORF_temp_mean', 'DUSSELDORF_temp_m
in',
       'DUSSELDORF_temp_max', 'HEATHROW_cloud_cover', 'HEATHROW_humidit
y',
       'HEATHROW_pressure', 'HEATHROW_global_radiation',
       'HEATHROW_precipitation', 'HEATHROW_sunshine', 'HEATHROW_temp_mea
n',
       'HEATHROW_temp_min', 'HEATHROW_temp_max', 'KASSEL_humidity',
       'KASSEL_pressure', 'KASSEL_global_radiation', 'KASSEL_precipitati
on',
       'KASSEL_sunshine', 'KASSEL_temp_mean', 'KASSEL_temp_min',
       'KASSEL_temp_max', 'MAASTRICHT_cloud_cover', 'MAASTRICHT_humidit
y',
       'MAASTRICHT_pressure', 'MAASTRICHT_global_radiation',
       'MAASTRICHT_precipitation', 'MAASTRICHT_sunshine',
       'MAASTRICHT_temp_mean', 'MAASTRICHT_temp_min', 'MAASTRICHT_temp_m
ax',
       'MALMO_precipitation', 'MALMO_temp_mean', 'MALMO_temp_min',
       'MALMO_temp_max', 'MUENCHEN_cloud_cover', 'MUENCHEN_humidity',
       'MUENCHEN_pressure', 'MUENCHEN_global_radiation',
       'MUENCHEN_precipitation', 'MUENCHEN_sunshine', 'MUENCHEN_temp_mea
n',
       'MUENCHEN_temp_min', 'MUENCHEN_temp_max', 'SONNBLICK_cloud_cove
r',
       'SONNBLICK_humidity', 'SONNBLICK_global_radiation',
       'SONNBLICK_precipitation', 'SONNBLICK_sunshine', 'SONNBLICK_temp_
mean',
       'SONNBLICK_temp_min', 'SONNBLICK_temp_max', 'TOURS_humidity',
       'TOURS_pressure', 'TOURS_global_radiation', 'TOURS_precipitatio
n',
       'TOURS_temp_mean', 'TOURS_temp_min', 'TOURS_temp_max'],
      dtype='object')
```

In [35]:
```
data.shape
```

Out[35]:
```
(3654, 91)
```

In [36]:
```
nr_rows = 365*3 # 3 years
# data
X_data = data.loc[:nr_rows] # Select first 3 years
X_data = X_data.drop(columns=['DATE', 'MONTH']) # Drop date and month col

# labels (sunshine hours the next day)
y_data = data.loc[1:(nr_rows + 1)]["BASEL_sunshine"]
```

In [37]:
```
from sklearn.model_selection import train_test_split

X_train, X_holdout, y_train, y_holdout = train_test_split(X_data, y_data,
```

In [38]:
```
X_val, X_test, y_val, y_test = train_test_split(X_holdout, y_holdout, tes
```

In [39]:
```
from tensorflow import keras

def create_nn():
    # Input layer
    inputs = keras.Input(shape=(X_data.shape[1],), name='input')

    # Dense layers
    layers_dense = keras.layers.Dense(100, 'relu')(inputs)
    layers_dense = keras.layers.Dense(50, 'relu')(layers_dense)

    # Output layer
    outputs = keras.layers.Dense(1)(layers_dense)

    return keras.Model(inputs=inputs, outputs=outputs, name="weather_pred

model = create_nn()
```

In [40]:
```
model.summary()
```

Model: "weather_prediction_model"

Layer (type)	Output Shape	
input (InputLayer)	(None, 89)	
dense_2 (Dense)	(None, 100)	
dense_3 (Dense)	(None, 50)	
dense_4 (Dense)	(None, 1)	

Total params: 14,101 (55.08 KB)
Trainable params: 14,101 (55.08 KB)
Non-trainable params: 0 (0.00 B)

In [41]:
```
model.compile(loss='mse')
```

In [42]:
```
model.compile(optimizer='adam',
              loss='mse')
```

In [43]:
```
model.compile(optimizer='adam',
              loss='mse',
              metrics=[keras.metrics.RootMeanSquaredError()])
```

In [44]:
```
def compile_model(model):
    model.compile(optimizer='adam',
                  loss='mse',
                  metrics=[keras.metrics.RootMeanSquaredError()])
compile_model(model)
```

In [45]:
```
history = model.fit(X_train, y_train,
                    batch_size=32,
                    epochs=200,
                    verbose=2)
```

```
Epoch 1/200
24/24 - 0s - 11ms/step - loss: 18.9691 - root_mean_squared_error: 4.3554
Epoch 2/200
24/24 - 0s - 525us/step - loss: 13.3472 - root_mean_squared_error: 3.6534
Epoch 3/200
24/24 - 0s - 596us/step - loss: 12.2244 - root_mean_squared_error: 3.4963
Epoch 4/200
24/24 - 0s - 584us/step - loss: 11.8429 - root_mean_squared_error: 3.4413
Epoch 5/200
24/24 - 0s - 553us/step - loss: 11.4458 - root_mean_squared_error: 3.3832
Epoch 6/200
24/24 - 0s - 568us/step - loss: 11.1932 - root_mean_squared_error: 3.3456
Epoch 7/200
24/24 - 0s - 602us/step - loss: 10.9514 - root_mean_squared_error: 3.3093
Epoch 8/200
24/24 - 0s - 629us/step - loss: 10.6909 - root_mean_squared_error: 3.2697
Epoch 9/200
24/24 - 0s - 621us/step - loss: 10.4790 - root_mean_squared_error: 3.2371
Epoch 10/200
24/24 - 0s - 562us/step - loss: 10.2942 - root_mean_squared_error: 3.2085
Epoch 11/200
24/24 - 0s - 574us/step - loss: 10.1918 - root_mean_squared_error: 3.1925
Epoch 12/200
24/24 - 0s - 589us/step - loss: 10.0514 - root_mean_squared_error: 3.1704
Epoch 13/200
24/24 - 0s - 580us/step - loss: 9.8657 - root_mean_squared_error: 3.1410
Epoch 14/200
24/24 - 0s - 585us/step - loss: 9.7928 - root_mean_squared_error: 3.1293
Epoch 15/200
24/24 - 0s - 569us/step - loss: 9.6197 - root_mean_squared_error: 3.1016
Epoch 16/200
24/24 - 0s - 572us/step - loss: 9.5327 - root_mean_squared_error: 3.0875
Epoch 17/200
24/24 - 0s - 570us/step - loss: 9.3993 - root_mean_squared_error: 3.0658
Epoch 18/200
24/24 - 0s - 551us/step - loss: 9.2256 - root_mean_squared_error: 3.0374
Epoch 19/200
24/24 - 0s - 565us/step - loss: 9.1806 - root_mean_squared_error: 3.0299
Epoch 20/200
24/24 - 0s - 1ms/step - loss: 9.0434 - root_mean_squared_error: 3.0072
Epoch 21/200
24/24 - 0s - 603us/step - loss: 8.9120 - root_mean_squared_error: 2.9853
Epoch 22/200
24/24 - 0s - 574us/step - loss: 8.8147 - root_mean_squared_error: 2.9690
Epoch 23/200
24/24 - 0s - 562us/step - loss: 8.7118 - root_mean_squared_error: 2.9516
Epoch 24/200
24/24 - 0s - 578us/step - loss: 8.5050 - root_mean_squared_error: 2.9163
Epoch 25/200
24/24 - 0s - 564us/step - loss: 8.3757 - root_mean_squared_error: 2.8941
Epoch 26/200
24/24 - 0s - 554us/step - loss: 8.1766 - root_mean_squared_error: 2.8595
Epoch 27/200
24/24 - 0s - 582us/step - loss: 8.1644 - root_mean_squared_error: 2.8573
Epoch 28/200
24/24 - 0s - 575us/step - loss: 8.1369 - root_mean_squared_error: 2.8525
Epoch 29/200
24/24 - 0s - 560us/step - loss: 8.1804 - root_mean_squared_error: 2.8601
Epoch 30/200
24/24 - 0s - 562us/step - loss: 7.8723 - root_mean_squared_error: 2.8058
```

```
Epoch 31/200
24/24 - 0s - 566us/step - loss: 7.8111 - root_mean_squared_error: 2.7948
Epoch 32/200
24/24 - 0s - 570us/step - loss: 7.5461 - root_mean_squared_error: 2.7470
Epoch 33/200
24/24 - 0s - 581us/step - loss: 7.4743 - root_mean_squared_error: 2.7339
Epoch 34/200
24/24 - 0s - 581us/step - loss: 7.3926 - root_mean_squared_error: 2.7189
Epoch 35/200
24/24 - 0s - 575us/step - loss: 7.2332 - root_mean_squared_error: 2.6895
Epoch 36/200
24/24 - 0s - 576us/step - loss: 7.2365 - root_mean_squared_error: 2.6901
Epoch 37/200
24/24 - 0s - 567us/step - loss: 7.1109 - root_mean_squared_error: 2.6666
Epoch 38/200
24/24 - 0s - 590us/step - loss: 7.0317 - root_mean_squared_error: 2.6517
Epoch 39/200
24/24 - 0s - 566us/step - loss: 6.7576 - root_mean_squared_error: 2.5995
Epoch 40/200
24/24 - 0s - 581us/step - loss: 6.9266 - root_mean_squared_error: 2.6318
Epoch 41/200
24/24 - 0s - 581us/step - loss: 6.8988 - root_mean_squared_error: 2.6266
Epoch 42/200
24/24 - 0s - 572us/step - loss: 6.9966 - root_mean_squared_error: 2.6451
Epoch 43/200
24/24 - 0s - 568us/step - loss: 6.8807 - root_mean_squared_error: 2.6231
Epoch 44/200
24/24 - 0s - 567us/step - loss: 6.7608 - root_mean_squared_error: 2.6002
Epoch 45/200
24/24 - 0s - 569us/step - loss: 6.7873 - root_mean_squared_error: 2.6052
Epoch 46/200
24/24 - 0s - 561us/step - loss: 6.7415 - root_mean_squared_error: 2.5964
Epoch 47/200
24/24 - 0s - 556us/step - loss: 6.7212 - root_mean_squared_error: 2.5925
Epoch 48/200
24/24 - 0s - 554us/step - loss: 6.6559 - root_mean_squared_error: 2.5799
Epoch 49/200
24/24 - 0s - 578us/step - loss: 6.5206 - root_mean_squared_error: 2.5536
Epoch 50/200
24/24 - 0s - 583us/step - loss: 6.3105 - root_mean_squared_error: 2.5121
Epoch 51/200
24/24 - 0s - 582us/step - loss: 6.0426 - root_mean_squared_error: 2.4582
Epoch 52/200
24/24 - 0s - 579us/step - loss: 5.8423 - root_mean_squared_error: 2.4171
Epoch 53/200
24/24 - 0s - 595us/step - loss: 5.6596 - root_mean_squared_error: 2.3790
Epoch 54/200
24/24 - 0s - 572us/step - loss: 5.5084 - root_mean_squared_error: 2.3470
Epoch 55/200
24/24 - 0s - 542us/step - loss: 5.3940 - root_mean_squared_error: 2.3225
Epoch 56/200
24/24 - 0s - 580us/step - loss: 5.3371 - root_mean_squared_error: 2.3102
Epoch 57/200
24/24 - 0s - 1ms/step - loss: 5.2570 - root_mean_squared_error: 2.2928
Epoch 58/200
24/24 - 0s - 547us/step - loss: 5.1782 - root_mean_squared_error: 2.2756
Epoch 59/200
24/24 - 0s - 601us/step - loss: 5.0941 - root_mean_squared_error: 2.2570
Epoch 60/200
24/24 - 0s - 561us/step - loss: 4.9979 - root_mean_squared_error: 2.2356
```

```
Epoch 61/200
24/24 - 0s - 564us/step - loss: 4.9081 - root_mean_squared_error: 2.2154
Epoch 62/200
24/24 - 0s - 578us/step - loss: 4.8427 - root_mean_squared_error: 2.2006
Epoch 63/200
24/24 - 0s - 577us/step - loss: 4.7299 - root_mean_squared_error: 2.1748
Epoch 64/200
24/24 - 0s - 567us/step - loss: 4.6046 - root_mean_squared_error: 2.1458
Epoch 65/200
24/24 - 0s - 548us/step - loss: 4.5324 - root_mean_squared_error: 2.1289
Epoch 66/200
24/24 - 0s - 603us/step - loss: 4.4329 - root_mean_squared_error: 2.1054
Epoch 67/200
24/24 - 0s - 583us/step - loss: 4.4189 - root_mean_squared_error: 2.1021
Epoch 68/200
24/24 - 0s - 589us/step - loss: 4.3663 - root_mean_squared_error: 2.0896
Epoch 69/200
24/24 - 0s - 573us/step - loss: 4.3117 - root_mean_squared_error: 2.0765
Epoch 70/200
24/24 - 0s - 578us/step - loss: 4.2282 - root_mean_squared_error: 2.0563
Epoch 71/200
24/24 - 0s - 577us/step - loss: 4.2159 - root_mean_squared_error: 2.0533
Epoch 72/200
24/24 - 0s - 580us/step - loss: 4.2002 - root_mean_squared_error: 2.0494
Epoch 73/200
24/24 - 0s - 553us/step - loss: 4.1115 - root_mean_squared_error: 2.0277
Epoch 74/200
24/24 - 0s - 548us/step - loss: 4.0043 - root_mean_squared_error: 2.0011
Epoch 75/200
24/24 - 0s - 571us/step - loss: 3.9981 - root_mean_squared_error: 1.9995
Epoch 76/200
24/24 - 0s - 572us/step - loss: 3.8635 - root_mean_squared_error: 1.9656
Epoch 77/200
24/24 - 0s - 597us/step - loss: 3.7608 - root_mean_squared_error: 1.9393
Epoch 78/200
24/24 - 0s - 573us/step - loss: 3.7356 - root_mean_squared_error: 1.9328
Epoch 79/200
24/24 - 0s - 585us/step - loss: 3.6473 - root_mean_squared_error: 1.9098
Epoch 80/200
24/24 - 0s - 561us/step - loss: 3.6804 - root_mean_squared_error: 1.9184
Epoch 81/200
24/24 - 0s - 567us/step - loss: 3.5599 - root_mean_squared_error: 1.8868
Epoch 82/200
24/24 - 0s - 555us/step - loss: 3.5148 - root_mean_squared_error: 1.8748
Epoch 83/200
24/24 - 0s - 556us/step - loss: 3.4517 - root_mean_squared_error: 1.8579
Epoch 84/200
24/24 - 0s - 583us/step - loss: 3.3314 - root_mean_squared_error: 1.8252
Epoch 85/200
24/24 - 0s - 578us/step - loss: 3.2962 - root_mean_squared_error: 1.8155
Epoch 86/200
24/24 - 0s - 565us/step - loss: 3.2305 - root_mean_squared_error: 1.7974
Epoch 87/200
24/24 - 0s - 589us/step - loss: 3.1915 - root_mean_squared_error: 1.7865
Epoch 88/200
24/24 - 0s - 582us/step - loss: 3.1227 - root_mean_squared_error: 1.7671
Epoch 89/200
24/24 - 0s - 572us/step - loss: 3.0797 - root_mean_squared_error: 1.7549
Epoch 90/200
24/24 - 0s - 1ms/step - loss: 3.0437 - root_mean_squared_error: 1.7446
```

```
Epoch 91/200
24/24 - 0s - 689us/step - loss: 3.0458 - root_mean_squared_error: 1.7452
Epoch 92/200
24/24 - 0s - 585us/step - loss: 3.2403 - root_mean_squared_error: 1.8001
Epoch 93/200
24/24 - 0s - 568us/step - loss: 3.2018 - root_mean_squared_error: 1.7894
Epoch 94/200
24/24 - 0s - 572us/step - loss: 3.0849 - root_mean_squared_error: 1.7564
Epoch 95/200
24/24 - 0s - 571us/step - loss: 3.0475 - root_mean_squared_error: 1.7457
Epoch 96/200
24/24 - 0s - 555us/step - loss: 2.9965 - root_mean_squared_error: 1.7310
Epoch 97/200
24/24 - 0s - 570us/step - loss: 2.9601 - root_mean_squared_error: 1.7205
Epoch 98/200
24/24 - 0s - 572us/step - loss: 3.0159 - root_mean_squared_error: 1.7366
Epoch 99/200
24/24 - 0s - 549us/step - loss: 2.8932 - root_mean_squared_error: 1.7010
Epoch 100/200
24/24 - 0s - 565us/step - loss: 2.9283 - root_mean_squared_error: 1.7112
Epoch 101/200
24/24 - 0s - 586us/step - loss: 3.0289 - root_mean_squared_error: 1.7404
Epoch 102/200
24/24 - 0s - 569us/step - loss: 3.1981 - root_mean_squared_error: 1.7883
Epoch 103/200
24/24 - 0s - 575us/step - loss: 3.2810 - root_mean_squared_error: 1.8113
Epoch 104/200
24/24 - 0s - 606us/step - loss: 3.3716 - root_mean_squared_error: 1.8362
Epoch 105/200
24/24 - 0s - 580us/step - loss: 3.6104 - root_mean_squared_error: 1.9001
Epoch 106/200
24/24 - 0s - 575us/step - loss: 3.9155 - root_mean_squared_error: 1.9788
Epoch 107/200
24/24 - 0s - 579us/step - loss: 3.8724 - root_mean_squared_error: 1.9678
Epoch 108/200
24/24 - 0s - 565us/step - loss: 3.9432 - root_mean_squared_error: 1.9857
Epoch 109/200
24/24 - 0s - 569us/step - loss: 3.9562 - root_mean_squared_error: 1.9890
Epoch 110/200
24/24 - 0s - 569us/step - loss: 3.6154 - root_mean_squared_error: 1.9014
Epoch 111/200
24/24 - 0s - 582us/step - loss: 3.4157 - root_mean_squared_error: 1.8482
Epoch 112/200
24/24 - 0s - 573us/step - loss: 3.2195 - root_mean_squared_error: 1.7943
Epoch 113/200
24/24 - 0s - 577us/step - loss: 3.0869 - root_mean_squared_error: 1.7570
Epoch 114/200
24/24 - 0s - 575us/step - loss: 3.0560 - root_mean_squared_error: 1.7481
Epoch 115/200
24/24 - 0s - 561us/step - loss: 2.8441 - root_mean_squared_error: 1.6865
Epoch 116/200
24/24 - 0s - 554us/step - loss: 2.8009 - root_mean_squared_error: 1.6736
Epoch 117/200
24/24 - 0s - 559us/step - loss: 3.0263 - root_mean_squared_error: 1.7396
Epoch 118/200
24/24 - 0s - 1ms/step - loss: 2.9198 - root_mean_squared_error: 1.7087
Epoch 119/200
24/24 - 0s - 559us/step - loss: 3.0377 - root_mean_squared_error: 1.7429
Epoch 120/200
24/24 - 0s - 583us/step - loss: 3.1106 - root_mean_squared_error: 1.7637
```

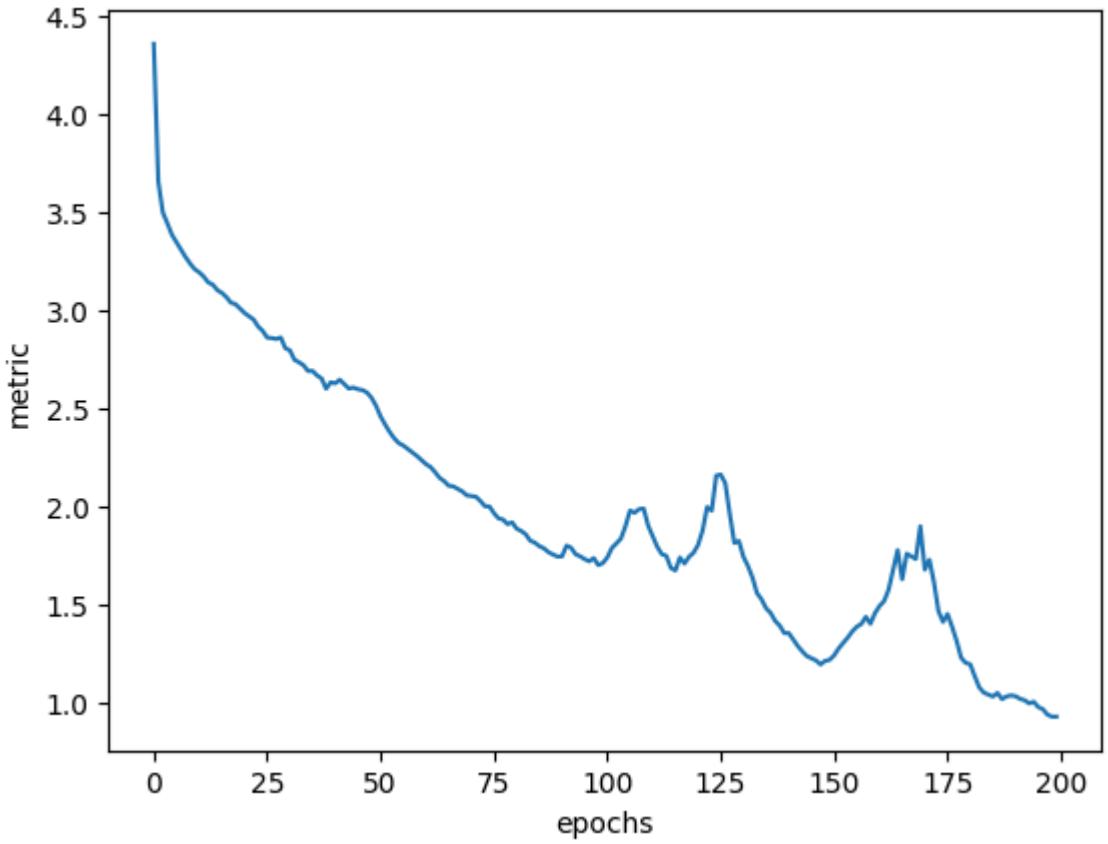
```
Epoch 121/200
24/24 - 0s - 569us/step - loss: 3.2533 - root_mean_squared_error: 1.8037
Epoch 122/200
24/24 - 0s - 620us/step - loss: 3.5250 - root_mean_squared_error: 1.8775
Epoch 123/200
24/24 - 0s - 573us/step - loss: 3.9873 - root_mean_squared_error: 1.9968
Epoch 124/200
24/24 - 0s - 587us/step - loss: 3.9129 - root_mean_squared_error: 1.9781
Epoch 125/200
24/24 - 0s - 569us/step - loss: 4.6500 - root_mean_squared_error: 2.1564
Epoch 126/200
24/24 - 0s - 577us/step - loss: 4.6738 - root_mean_squared_error: 2.1619
Epoch 127/200
24/24 - 0s - 555us/step - loss: 4.4842 - root_mean_squared_error: 2.1176
Epoch 128/200
24/24 - 0s - 596us/step - loss: 3.8409 - root_mean_squared_error: 1.9598
Epoch 129/200
24/24 - 0s - 563us/step - loss: 3.2892 - root_mean_squared_error: 1.8136
Epoch 130/200
24/24 - 0s - 569us/step - loss: 3.3283 - root_mean_squared_error: 1.8244
Epoch 131/200
24/24 - 0s - 573us/step - loss: 3.0386 - root_mean_squared_error: 1.7432
Epoch 132/200
24/24 - 0s - 575us/step - loss: 2.8774 - root_mean_squared_error: 1.6963
Epoch 133/200
24/24 - 0s - 580us/step - loss: 2.6803 - root_mean_squared_error: 1.6372
Epoch 134/200
24/24 - 0s - 568us/step - loss: 2.4274 - root_mean_squared_error: 1.5580
Epoch 135/200
24/24 - 0s - 579us/step - loss: 2.3299 - root_mean_squared_error: 1.5264
Epoch 136/200
24/24 - 0s - 561us/step - loss: 2.1936 - root_mean_squared_error: 1.4811
Epoch 137/200
24/24 - 0s - 571us/step - loss: 2.1236 - root_mean_squared_error: 1.4572
Epoch 138/200
24/24 - 0s - 578us/step - loss: 2.0074 - root_mean_squared_error: 1.4168
Epoch 139/200
24/24 - 0s - 575us/step - loss: 1.9401 - root_mean_squared_error: 1.3929
Epoch 140/200
24/24 - 0s - 575us/step - loss: 1.8355 - root_mean_squared_error: 1.3548
Epoch 141/200
24/24 - 0s - 586us/step - loss: 1.8353 - root_mean_squared_error: 1.3547
Epoch 142/200
24/24 - 0s - 1ms/step - loss: 1.7440 - root_mean_squared_error: 1.3206
Epoch 143/200
24/24 - 0s - 615us/step - loss: 1.6594 - root_mean_squared_error: 1.2882
Epoch 144/200
24/24 - 0s - 576us/step - loss: 1.5903 - root_mean_squared_error: 1.2611
Epoch 145/200
24/24 - 0s - 564us/step - loss: 1.5313 - root_mean_squared_error: 1.2375
Epoch 146/200
24/24 - 0s - 561us/step - loss: 1.5048 - root_mean_squared_error: 1.2267
Epoch 147/200
24/24 - 0s - 569us/step - loss: 1.4770 - root_mean_squared_error: 1.2153
Epoch 148/200
24/24 - 0s - 574us/step - loss: 1.4283 - root_mean_squared_error: 1.1951
Epoch 149/200
24/24 - 0s - 565us/step - loss: 1.4711 - root_mean_squared_error: 1.2129
Epoch 150/200
24/24 - 0s - 586us/step - loss: 1.4845 - root_mean_squared_error: 1.2184
```

```
Epoch 151/200
24/24 - 0s - 584us/step - loss: 1.5410 - root_mean_squared_error: 1.2414
Epoch 152/200
24/24 - 0s - 576us/step - loss: 1.6304 - root_mean_squared_error: 1.2769
Epoch 153/200
24/24 - 0s - 559us/step - loss: 1.7023 - root_mean_squared_error: 1.3047
Epoch 154/200
24/24 - 0s - 554us/step - loss: 1.7743 - root_mean_squared_error: 1.3320
Epoch 155/200
24/24 - 0s - 565us/step - loss: 1.8584 - root_mean_squared_error: 1.3632
Epoch 156/200
24/24 - 0s - 571us/step - loss: 1.9242 - root_mean_squared_error: 1.3872
Epoch 157/200
24/24 - 0s - 581us/step - loss: 1.9646 - root_mean_squared_error: 1.4016
Epoch 158/200
24/24 - 0s - 574us/step - loss: 2.0662 - root_mean_squared_error: 1.4374
Epoch 159/200
24/24 - 0s - 578us/step - loss: 1.9669 - root_mean_squared_error: 1.4025
Epoch 160/200
24/24 - 0s - 587us/step - loss: 2.1212 - root_mean_squared_error: 1.4564
Epoch 161/200
24/24 - 0s - 577us/step - loss: 2.2271 - root_mean_squared_error: 1.4924
Epoch 162/200
24/24 - 0s - 577us/step - loss: 2.3020 - root_mean_squared_error: 1.5172
Epoch 163/200
24/24 - 0s - 570us/step - loss: 2.4823 - root_mean_squared_error: 1.5755
Epoch 164/200
24/24 - 0s - 718us/step - loss: 2.8142 - root_mean_squared_error: 1.6776
Epoch 165/200
24/24 - 0s - 583us/step - loss: 3.1598 - root_mean_squared_error: 1.7776
Epoch 166/200
24/24 - 0s - 573us/step - loss: 2.6538 - root_mean_squared_error: 1.6290
Epoch 167/200
24/24 - 0s - 572us/step - loss: 3.0936 - root_mean_squared_error: 1.7589
Epoch 168/200
24/24 - 0s - 592us/step - loss: 3.0473 - root_mean_squared_error: 1.7456
Epoch 169/200
24/24 - 0s - 580us/step - loss: 3.0009 - root_mean_squared_error: 1.7323
Epoch 170/200
24/24 - 0s - 584us/step - loss: 3.6061 - root_mean_squared_error: 1.8990
Epoch 171/200
24/24 - 0s - 583us/step - loss: 2.8146 - root_mean_squared_error: 1.6777
Epoch 172/200
24/24 - 0s - 575us/step - loss: 2.9859 - root_mean_squared_error: 1.7280
Epoch 173/200
24/24 - 0s - 562us/step - loss: 2.6074 - root_mean_squared_error: 1.6147
Epoch 174/200
24/24 - 0s - 580us/step - loss: 2.1512 - root_mean_squared_error: 1.4667
Epoch 175/200
24/24 - 0s - 582us/step - loss: 1.9928 - root_mean_squared_error: 1.4117
Epoch 176/200
24/24 - 0s - 590us/step - loss: 2.1081 - root_mean_squared_error: 1.4519
Epoch 177/200
24/24 - 0s - 574us/step - loss: 1.9247 - root_mean_squared_error: 1.3873
Epoch 178/200
24/24 - 0s - 583us/step - loss: 1.7293 - root_mean_squared_error: 1.3150
Epoch 179/200
24/24 - 0s - 575us/step - loss: 1.5110 - root_mean_squared_error: 1.2292
Epoch 180/200
24/24 - 0s - 582us/step - loss: 1.4476 - root_mean_squared_error: 1.2032
```

```
Epoch 181/200
24/24 - 0s - 568us/step - loss: 1.4303 - root_mean_squared_error: 1.1959
Epoch 182/200
24/24 - 0s - 552us/step - loss: 1.2883 - root_mean_squared_error: 1.1350
Epoch 183/200
24/24 - 0s - 1ms/step - loss: 1.1619 - root_mean_squared_error: 1.0779
Epoch 184/200
24/24 - 0s - 559us/step - loss: 1.1058 - root_mean_squared_error: 1.0516
Epoch 185/200
24/24 - 0s - 569us/step - loss: 1.0844 - root_mean_squared_error: 1.0413
Epoch 186/200
24/24 - 0s - 568us/step - loss: 1.0642 - root_mean_squared_error: 1.0316
Epoch 187/200
24/24 - 0s - 570us/step - loss: 1.1026 - root_mean_squared_error: 1.0501
Epoch 188/200
24/24 - 0s - 560us/step - loss: 1.0375 - root_mean_squared_error: 1.0186
Epoch 189/200
24/24 - 0s - 582us/step - loss: 1.0649 - root_mean_squared_error: 1.0319
Epoch 190/200
24/24 - 0s - 582us/step - loss: 1.0765 - root_mean_squared_error: 1.0376
Epoch 191/200
24/24 - 0s - 576us/step - loss: 1.0675 - root_mean_squared_error: 1.0332
Epoch 192/200
24/24 - 0s - 550us/step - loss: 1.0401 - root_mean_squared_error: 1.0199
Epoch 193/200
24/24 - 0s - 580us/step - loss: 1.0256 - root_mean_squared_error: 1.0127
Epoch 194/200
24/24 - 0s - 612us/step - loss: 0.9932 - root_mean_squared_error: 0.9966
Epoch 195/200
24/24 - 0s - 591us/step - loss: 1.0103 - root_mean_squared_error: 1.0052
Epoch 196/200
24/24 - 0s - 570us/step - loss: 0.9560 - root_mean_squared_error: 0.9777
Epoch 197/200
24/24 - 0s - 592us/step - loss: 0.9364 - root_mean_squared_error: 0.9677
Epoch 198/200
24/24 - 0s - 583us/step - loss: 0.8832 - root_mean_squared_error: 0.9398
Epoch 199/200
24/24 - 0s - 580us/step - loss: 0.8634 - root_mean_squared_error: 0.9292
Epoch 200/200
24/24 - 0s - 565us/step - loss: 0.8632 - root_mean_squared_error: 0.9291
```

In [46]:

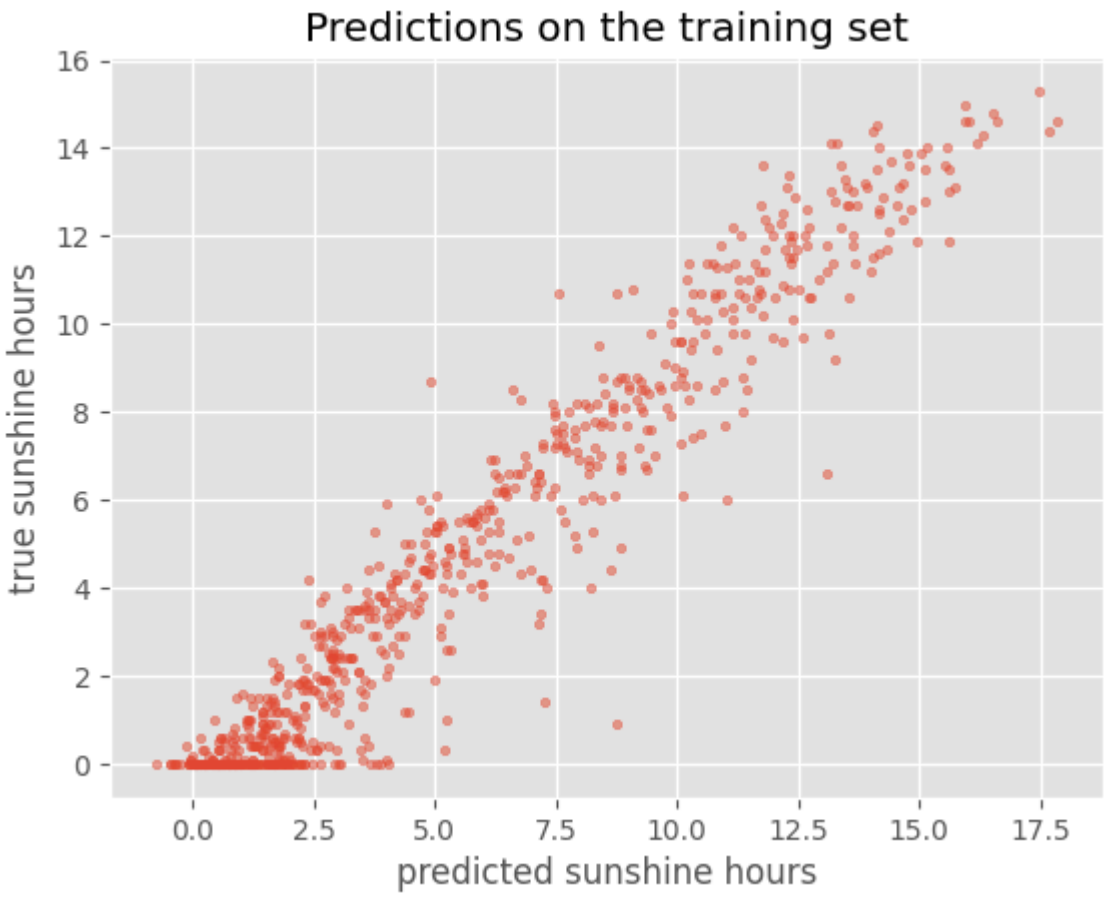
```python
import seaborn as sns
import matplotlib.pyplot as plt

def plot_history(history, metrics):
    """
    Plot the training history

    Args:
        history (keras History object that is returned by model.fit())
        metrics (str, list): Metric or a list of metrics to plot
    """
    history_df = pd.DataFrame.from_dict(history.history)
    sns.lineplot(data=history_df[metrics])
    plt.xlabel("epochs")
    plt.ylabel("metric")

plot_history(history, 'root_mean_squared_error')
```



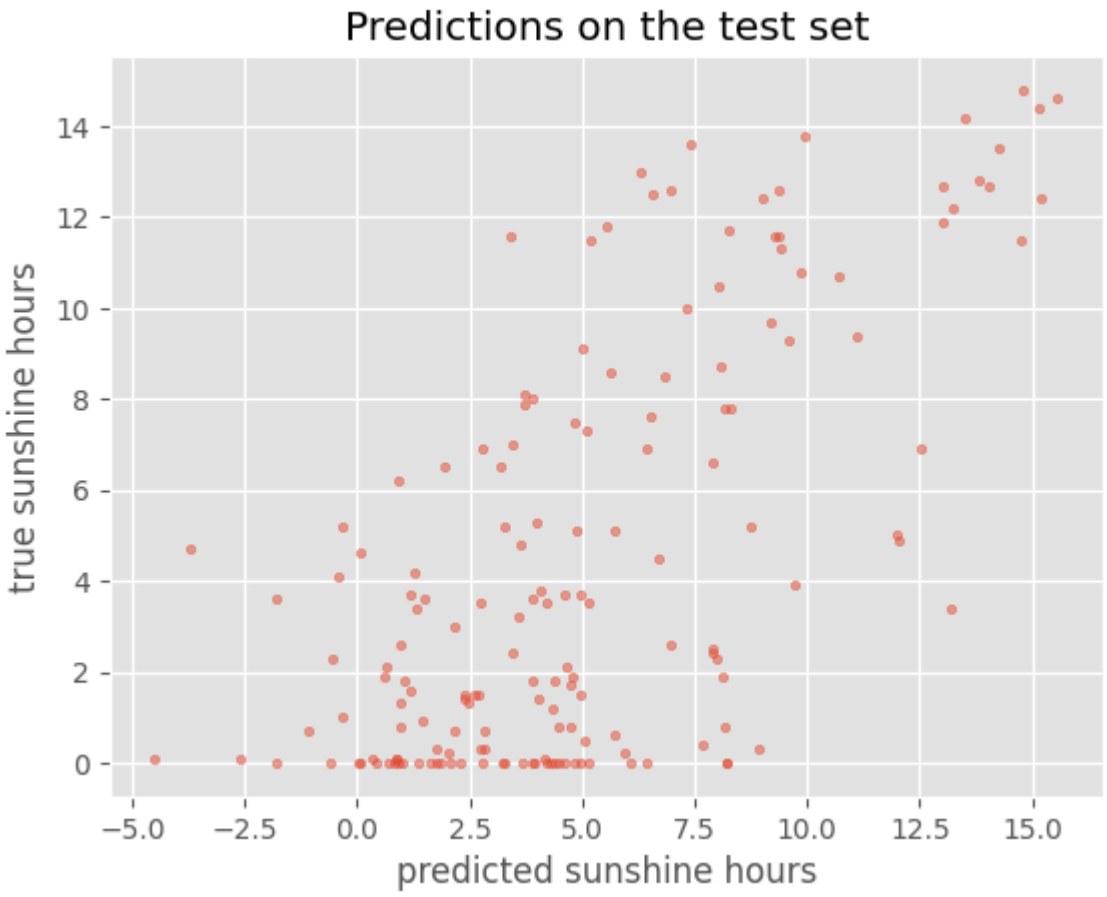
In [47]:
```
y_train_predicted = model.predict(X_train)
y_test_predicted = model.predict(X_test)
```

```
WARNING:tensorflow:5 out of the last 7 calls to <function TensorFlowTraine
r.make_predict_function.<locals>.one_step_on_data_distributed at 0x16bb5da
80> triggered tf.function retracing. Tracing is expensive and the excessiv
e number of tracings could be due to (1) creating @tf.function repeatedly
in a loop, (2) passing tensors with different shapes, (3) passing Python o
bjects instead of tensors. For (1), please define your @tf.function outsid
e of the loop. For (2), @tf.function has reduce_retracing=True option that
can avoid unnecessary retracing. For (3), please refer to https://www.tens
orflow.org/guide/function#controlling_retracing and https://www.tensorflo
w.org/api_docs/python/tf/function for  more details.
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 811us/step
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 356us/step
```

In [48]:
```
# We define a function that we will reuse in this lesson
def plot_predictions(y_pred, y_true, title):
    plt.style.use('ggplot')  # optional, that's only to define a visual s
    plt.scatter(y_pred, y_true, s=10, alpha=0.5)
    plt.xlabel("predicted sunshine hours")
    plt.ylabel("true sunshine hours")
    plt.title(title)

plot_predictions(y_train_predicted, y_train, title='Predictions on the tr
```



In [49]:
```
plot_predictions(y_test_predicted, y_test, title='Predictions on the test
```

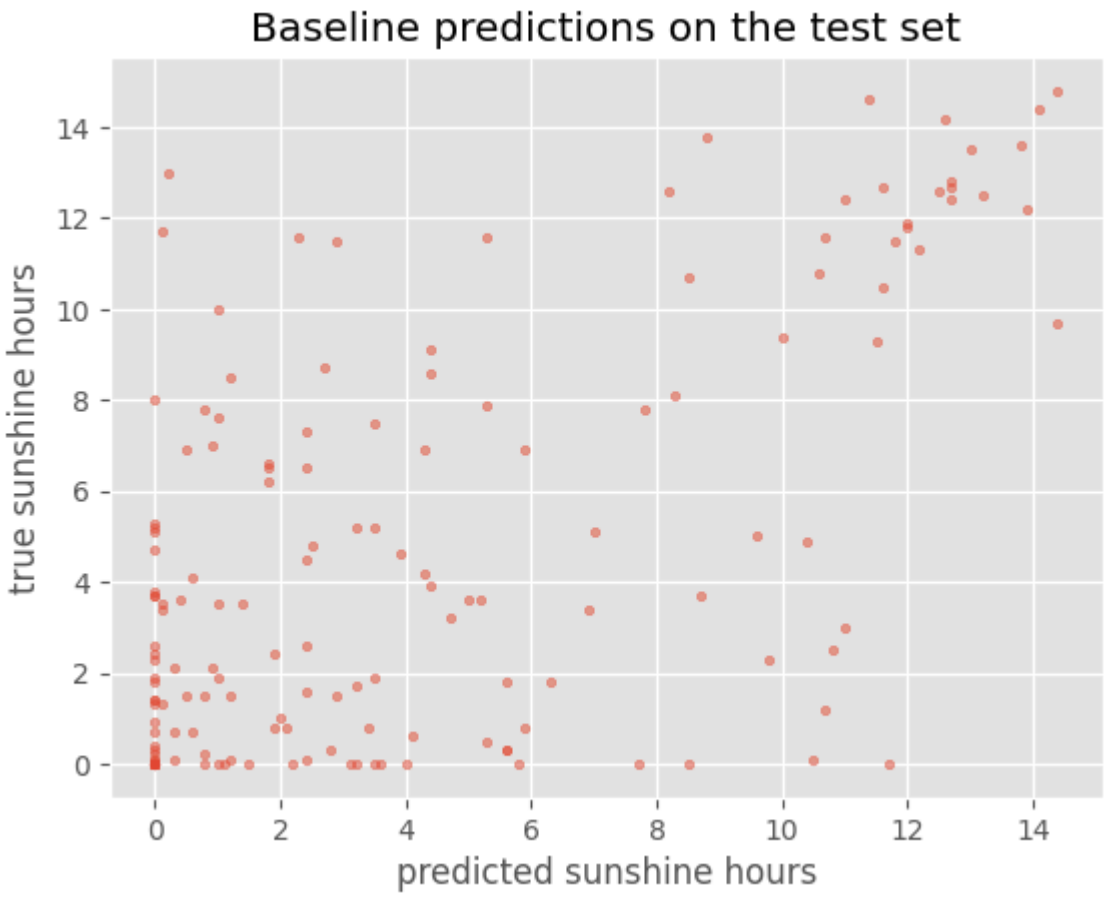


In [50]:
```
train_metrics = model.evaluate(X_train, y_train, return_dict=True)
test_metrics = model.evaluate(X_test, y_test, return_dict=True)
```

```
print('Train RMSE: {:.2f}, Test RMSE: {:.2f}'.format(train_metrics['root_
```

```
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 274us/step - loss: 2.0585 - root_mean_square
d_error: 1.4340
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 450us/step - loss: 13.9154 - root_mean_squared
_error: 3.7272
Train RMSE: 1.45, Test RMSE: 3.62
```

In [51]:
```
y_baseline_prediction = X_test['BASEL_sunshine']
plot_predictions(y_baseline_prediction, y_test, title='Baseline predictio
```



In [52]:
```
from sklearn.metrics import mean_squared_error
rmse_baseline = mean_squared_error(y_test, y_baseline_prediction, squared
print('Baseline:', rmse_baseline)
print('Neural network: ', test_metrics['root_mean_squared_error'])
```

```
Baseline: 3.877323350410224
Neural network:  3.6203036308288574
```

```
/Users/johannabayer/Documents/dl_workshop/lib/python3.11/site-packages/skl
earn/metrics/_regression.py:492: FutureWarning: 'squared' is deprecated in
version 1.4 and will be removed in 1.6. To calculate the root mean squared
error, use the function'root_mean_squared_error'.
  warnings.warn(
```

In [53]:
```
model = create_nn()
compile_model(model)
```

In [54]:
```
history = model.fit(X_train, y_train,
                    batch_size=32,
                    epochs=200,
                    validation_data=(X_val, y_val))
```

```
Epoch 1/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 19.2158 - root_mean_squared
_error: 4.3689 - val_loss: 12.0585 - val_root_mean_squared_error: 3.4725
Epoch 2/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 630us/step - loss: 13.1236 - root_mean_squar
ed_error: 3.6201 - val_loss: 12.3589 - val_root_mean_squared_error: 3.5155
Epoch 3/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 793us/step - loss: 12.2215 - root_mean_squar
ed_error: 3.4937 - val_loss: 12.6999 - val_root_mean_squared_error: 3.5637
Epoch 4/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 710us/step - loss: 11.7716 - root_mean_squar
ed_error: 3.4287 - val_loss: 12.9874 - val_root_mean_squared_error: 3.6038
Epoch 5/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 677us/step - loss: 11.6700 - root_mean_squar
ed_error: 3.4138 - val_loss: 13.0512 - val_root_mean_squared_error: 3.6126
Epoch 6/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 675us/step - loss: 11.7592 - root_mean_squar
ed_error: 3.4267 - val_loss: 13.2446 - val_root_mean_squared_error: 3.6393
Epoch 7/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 708us/step - loss: 12.0380 - root_mean_squar
ed_error: 3.4677 - val_loss: 12.1305 - val_root_mean_squared_error: 3.4829
Epoch 8/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 692us/step - loss: 12.4490 - root_mean_squar
ed_error: 3.5270 - val_loss: 10.8244 - val_root_mean_squared_error: 3.2900
Epoch 9/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 675us/step - loss: 12.2921 - root_mean_squar
ed_error: 3.5039 - val_loss: 10.7070 - val_root_mean_squared_error: 3.2722
Epoch 10/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 674us/step - loss: 11.2640 - root_mean_squar
ed_error: 3.3494 - val_loss: 10.5880 - val_root_mean_squared_error: 3.2539
Epoch 11/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 701us/step - loss: 10.7867 - root_mean_squar
ed_error: 3.2806 - val_loss: 10.5052 - val_root_mean_squared_error: 3.2412
Epoch 12/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 669us/step - loss: 10.8997 - root_mean_squar
ed_error: 3.2974 - val_loss: 10.4995 - val_root_mean_squared_error: 3.2403
Epoch 13/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 676us/step - loss: 10.6770 - root_mean_squar
ed_error: 3.2622 - val_loss: 10.3196 - val_root_mean_squared_error: 3.2124
Epoch 14/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 687us/step - loss: 10.3374 - root_mean_squar
ed_error: 3.2106 - val_loss: 10.5893 - val_root_mean_squared_error: 3.2541
Epoch 15/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 683us/step - loss: 10.1556 - root_mean_squar
ed_error: 3.1827 - val_loss: 10.3647 - val_root_mean_squared_error: 3.2194
Epoch 16/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 687us/step - loss: 9.9719 - root_mean_square
d_error: 3.1538 - val_loss: 10.4376 - val_root_mean_squared_error: 3.2307
Epoch 17/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 666us/step - loss: 9.7312 - root_mean_square
d_error: 3.1161 - val_loss: 10.4057 - val_root_mean_squared_error: 3.2258
Epoch 18/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 664us/step - loss: 9.5503 - root_mean_square
d_error: 3.0870 - val_loss: 10.3325 - val_root_mean_squared_error: 3.2144
Epoch 19/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 655us/step - loss: 9.4619 - root_mean_square
d_error: 3.0723 - val_loss: 10.2296 - val_root_mean_squared_error: 3.1984
Epoch 20/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 691us/step - loss: 9.2155 - root_mean_square
d_error: 3.0321 - val_loss: 10.2949 - val_root_mean_squared_error: 3.2086
```

```
Epoch 21/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 679us/step - loss: 9.0251 - root_mean_square
d_error: 3.0004 - val_loss: 10.2357 - val_root_mean_squared_error: 3.1993
Epoch 22/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 663us/step - loss: 8.8773 - root_mean_square
d_error: 2.9757 - val_loss: 10.4722 - val_root_mean_squared_error: 3.2361
Epoch 23/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 705us/step - loss: 8.6956 - root_mean_square
d_error: 2.9456 - val_loss: 10.3696 - val_root_mean_squared_error: 3.2202
Epoch 24/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 661us/step - loss: 8.5848 - root_mean_square
d_error: 2.9270 - val_loss: 10.3793 - val_root_mean_squared_error: 3.2217
Epoch 25/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 664us/step - loss: 8.4718 - root_mean_square
d_error: 2.9074 - val_loss: 10.6015 - val_root_mean_squared_error: 3.2560
Epoch 26/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 684us/step - loss: 8.4667 - root_mean_square
d_error: 2.9066 - val_loss: 10.4999 - val_root_mean_squared_error: 3.2404
Epoch 27/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 673us/step - loss: 8.2497 - root_mean_square
d_error: 2.8693 - val_loss: 10.5498 - val_root_mean_squared_error: 3.2480
Epoch 28/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 672us/step - loss: 8.2241 - root_mean_square
d_error: 2.8654 - val_loss: 10.4725 - val_root_mean_squared_error: 3.2361
Epoch 29/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 784us/step - loss: 8.2006 - root_mean_square
d_error: 2.8609 - val_loss: 10.2424 - val_root_mean_squared_error: 3.2004
Epoch 30/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 672us/step - loss: 7.9619 - root_mean_square
d_error: 2.8193 - val_loss: 10.2046 - val_root_mean_squared_error: 3.1945
Epoch 31/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 660us/step - loss: 7.8587 - root_mean_square
d_error: 2.8002 - val_loss: 10.2449 - val_root_mean_squared_error: 3.2008
Epoch 32/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 698us/step - loss: 7.6910 - root_mean_square
d_error: 2.7702 - val_loss: 10.1774 - val_root_mean_squared_error: 3.1902
Epoch 33/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 688us/step - loss: 7.5687 - root_mean_square
d_error: 2.7483 - val_loss: 10.3934 - val_root_mean_squared_error: 3.2239
Epoch 34/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 670us/step - loss: 7.3141 - root_mean_square
d_error: 2.7011 - val_loss: 10.4583 - val_root_mean_squared_error: 3.2339
Epoch 35/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 683us/step - loss: 7.1962 - root_mean_square
d_error: 2.6800 - val_loss: 10.8937 - val_root_mean_squared_error: 3.3006
Epoch 36/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 685us/step - loss: 6.7054 - root_mean_square
d_error: 2.5862 - val_loss: 10.8556 - val_root_mean_squared_error: 3.2948
Epoch 37/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 706us/step - loss: 6.8342 - root_mean_square
d_error: 2.6113 - val_loss: 10.9459 - val_root_mean_squared_error: 3.3085
Epoch 38/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 758us/step - loss: 6.5735 - root_mean_square
d_error: 2.5609 - val_loss: 10.8710 - val_root_mean_squared_error: 3.2971
Epoch 39/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 698us/step - loss: 6.4396 - root_mean_square
d_error: 2.5352 - val_loss: 10.9636 - val_root_mean_squared_error: 3.3111
Epoch 40/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 666us/step - loss: 6.3101 - root_mean_square
d_error: 2.5096 - val_loss: 11.1976 - val_root_mean_squared_error: 3.3463
```

```
Epoch 41/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 670us/step - loss: 6.2780 - root_mean_square
d_error: 2.5036 - val_loss: 10.9917 - val_root_mean_squared_error: 3.3154
Epoch 42/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 671us/step - loss: 6.3539 - root_mean_square
d_error: 2.5182 - val_loss: 11.1902 - val_root_mean_squared_error: 3.3452
Epoch 43/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 680us/step - loss: 6.3151 - root_mean_square
d_error: 2.5112 - val_loss: 10.9826 - val_root_mean_squared_error: 3.3140
Epoch 44/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 683us/step - loss: 6.2585 - root_mean_square
d_error: 2.4999 - val_loss: 11.1668 - val_root_mean_squared_error: 3.3417
Epoch 45/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 747us/step - loss: 6.2653 - root_mean_square
d_error: 2.5012 - val_loss: 11.0656 - val_root_mean_squared_error: 3.3265
Epoch 46/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 670us/step - loss: 6.2132 - root_mean_square
d_error: 2.4906 - val_loss: 11.0061 - val_root_mean_squared_error: 3.3175
Epoch 47/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 669us/step - loss: 5.9761 - root_mean_square
d_error: 2.4431 - val_loss: 11.1269 - val_root_mean_squared_error: 3.3357
Epoch 48/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 691us/step - loss: 5.8757 - root_mean_square
d_error: 2.4228 - val_loss: 11.2693 - val_root_mean_squared_error: 3.3570
Epoch 49/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 660us/step - loss: 5.9591 - root_mean_square
d_error: 2.4397 - val_loss: 11.2513 - val_root_mean_squared_error: 3.3543
Epoch 50/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 705us/step - loss: 5.7258 - root_mean_square
d_error: 2.3918 - val_loss: 11.7636 - val_root_mean_squared_error: 3.4298
Epoch 51/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 690us/step - loss: 5.5711 - root_mean_square
d_error: 2.3593 - val_loss: 11.7142 - val_root_mean_squared_error: 3.4226
Epoch 52/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 682us/step - loss: 5.4893 - root_mean_square
d_error: 2.3418 - val_loss: 11.2651 - val_root_mean_squared_error: 3.3564
Epoch 53/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 697us/step - loss: 5.4487 - root_mean_square
d_error: 2.3325 - val_loss: 11.8994 - val_root_mean_squared_error: 3.4496
Epoch 54/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 665us/step - loss: 5.1521 - root_mean_square
d_error: 2.2685 - val_loss: 12.0630 - val_root_mean_squared_error: 3.4732
Epoch 55/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 675us/step - loss: 5.0471 - root_mean_square
d_error: 2.2455 - val_loss: 11.8490 - val_root_mean_squared_error: 3.4422
Epoch 56/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 692us/step - loss: 4.9823 - root_mean_square
d_error: 2.2301 - val_loss: 12.1369 - val_root_mean_squared_error: 3.4838
Epoch 57/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 4.8073 - root_mean_squared_
error: 2.1915 - val_loss: 11.8775 - val_root_mean_squared_error: 3.4464
Epoch 58/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 682us/step - loss: 4.5872 - root_mean_square
d_error: 2.1408 - val_loss: 11.4342 - val_root_mean_squared_error: 3.3814
Epoch 59/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 685us/step - loss: 4.5994 - root_mean_square
d_error: 2.1438 - val_loss: 11.5086 - val_root_mean_squared_error: 3.3924
Epoch 60/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 675us/step - loss: 4.5200 - root_mean_square
d_error: 2.1251 - val_loss: 11.1254 - val_root_mean_squared_error: 3.3355
```

```
Epoch 61/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 685us/step - loss: 4.6807 - root_mean_square
d_error: 2.1626 - val_loss: 11.6270 - val_root_mean_squared_error: 3.4098
Epoch 62/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 691us/step - loss: 4.6002 - root_mean_square
d_error: 2.1441 - val_loss: 11.6846 - val_root_mean_squared_error: 3.4183
Epoch 63/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 698us/step - loss: 4.2471 - root_mean_square
d_error: 2.0600 - val_loss: 11.8044 - val_root_mean_squared_error: 3.4358
Epoch 64/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 703us/step - loss: 4.3109 - root_mean_square
d_error: 2.0757 - val_loss: 11.3812 - val_root_mean_squared_error: 3.3736
Epoch 65/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 678us/step - loss: 4.2927 - root_mean_square
d_error: 2.0711 - val_loss: 11.2915 - val_root_mean_squared_error: 3.3603
Epoch 66/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 670us/step - loss: 4.6051 - root_mean_square
d_error: 2.1453 - val_loss: 11.5351 - val_root_mean_squared_error: 3.3963
Epoch 67/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 4.4605 - root_mean_squared_
error: 2.1112 - val_loss: 11.4802 - val_root_mean_squared_error: 3.3882
Epoch 68/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 692us/step - loss: 4.3380 - root_mean_square
d_error: 2.0820 - val_loss: 11.6941 - val_root_mean_squared_error: 3.4197
Epoch 69/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 687us/step - loss: 3.9995 - root_mean_square
d_error: 1.9989 - val_loss: 12.0253 - val_root_mean_squared_error: 3.4678
Epoch 70/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 675us/step - loss: 3.8307 - root_mean_square
d_error: 1.9565 - val_loss: 12.2710 - val_root_mean_squared_error: 3.5030
Epoch 71/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 670us/step - loss: 3.9348 - root_mean_square
d_error: 1.9828 - val_loss: 12.5693 - val_root_mean_squared_error: 3.5453
Epoch 72/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 679us/step - loss: 3.7721 - root_mean_square
d_error: 1.9416 - val_loss: 12.7255 - val_root_mean_squared_error: 3.5673
Epoch 73/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 714us/step - loss: 3.6415 - root_mean_square
d_error: 1.9079 - val_loss: 12.4822 - val_root_mean_squared_error: 3.5330
Epoch 74/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 685us/step - loss: 3.6218 - root_mean_square
d_error: 1.9026 - val_loss: 12.2177 - val_root_mean_squared_error: 3.4954
Epoch 75/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 698us/step - loss: 3.4657 - root_mean_square
d_error: 1.8612 - val_loss: 12.4659 - val_root_mean_squared_error: 3.5307
Epoch 76/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3.2450 - root_mean_squared_
error: 1.8009 - val_loss: 12.1911 - val_root_mean_squared_error: 3.4916
Epoch 77/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 686us/step - loss: 3.2772 - root_mean_square
d_error: 1.8097 - val_loss: 12.7351 - val_root_mean_squared_error: 3.5686
Epoch 78/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 691us/step - loss: 3.1976 - root_mean_square
d_error: 1.7876 - val_loss: 12.9250 - val_root_mean_squared_error: 3.5951
Epoch 79/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 692us/step - loss: 3.1912 - root_mean_square
d_error: 1.7858 - val_loss: 12.8653 - val_root_mean_squared_error: 3.5868
Epoch 80/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 685us/step - loss: 3.1489 - root_mean_square
d_error: 1.7739 - val_loss: 12.7213 - val_root_mean_squared_error: 3.5667
```

```
Epoch 81/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 676us/step - loss: 2.9403 - root_mean_square
d_error: 1.7140 - val_loss: 13.0819 - val_root_mean_squared_error: 3.6169
Epoch 82/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 687us/step - loss: 2.9988 - root_mean_square
d_error: 1.7309 - val_loss: 13.0579 - val_root_mean_squared_error: 3.6136
Epoch 83/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 686us/step - loss: 2.9083 - root_mean_square
d_error: 1.7044 - val_loss: 13.2272 - val_root_mean_squared_error: 3.6369
Epoch 84/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 2.9170 - root_mean_squared_
error: 1.7070 - val_loss: 13.5285 - val_root_mean_squared_error: 3.6781
Epoch 85/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 744us/step - loss: 2.8420 - root_mean_square
d_error: 1.6852 - val_loss: 13.8845 - val_root_mean_squared_error: 3.7262
Epoch 86/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 722us/step - loss: 2.6874 - root_mean_square
d_error: 1.6387 - val_loss: 14.1019 - val_root_mean_squared_error: 3.7552
Epoch 87/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 724us/step - loss: 2.6564 - root_mean_square
d_error: 1.6291 - val_loss: 14.2976 - val_root_mean_squared_error: 3.7812
Epoch 88/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 686us/step - loss: 2.7343 - root_mean_square
d_error: 1.6529 - val_loss: 13.2317 - val_root_mean_squared_error: 3.6375
Epoch 89/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 713us/step - loss: 2.8352 - root_mean_square
d_error: 1.6815 - val_loss: 13.2614 - val_root_mean_squared_error: 3.6416
Epoch 90/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 687us/step - loss: 2.8244 - root_mean_square
d_error: 1.6773 - val_loss: 13.2090 - val_root_mean_squared_error: 3.6344
Epoch 91/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 664us/step - loss: 2.9485 - root_mean_square
d_error: 1.7139 - val_loss: 13.6281 - val_root_mean_squared_error: 3.6916
Epoch 92/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3.0014 - root_mean_squared_
error: 1.7304 - val_loss: 12.7879 - val_root_mean_squared_error: 3.5760
Epoch 93/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 691us/step - loss: 2.9780 - root_mean_square
d_error: 1.7245 - val_loss: 12.6029 - val_root_mean_squared_error: 3.5501
Epoch 94/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 760us/step - loss: 3.3933 - root_mean_square
d_error: 1.8413 - val_loss: 12.6297 - val_root_mean_squared_error: 3.5538
Epoch 95/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 695us/step - loss: 3.5881 - root_mean_square
d_error: 1.8938 - val_loss: 13.3788 - val_root_mean_squared_error: 3.6577
Epoch 96/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 679us/step - loss: 3.2135 - root_mean_square
d_error: 1.7914 - val_loss: 13.1674 - val_root_mean_squared_error: 3.6287
Epoch 97/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 674us/step - loss: 2.8930 - root_mean_square
d_error: 1.7005 - val_loss: 13.5202 - val_root_mean_squared_error: 3.6770
Epoch 98/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 676us/step - loss: 2.6186 - root_mean_square
d_error: 1.6178 - val_loss: 12.9336 - val_root_mean_squared_error: 3.5963
Epoch 99/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 2.5361 - root_mean_squared_
error: 1.5920 - val_loss: 13.3805 - val_root_mean_squared_error: 3.6579
Epoch 100/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 713us/step - loss: 2.3710 - root_mean_square
d_error: 1.5391 - val_loss: 13.1512 - val_root_mean_squared_error: 3.6265
```

```
Epoch 101/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 680us/step - loss: 2.5126 - root_mean_square
d_error: 1.5845 - val_loss: 13.2633 - val_root_mean_squared_error: 3.6419
Epoch 102/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 670us/step - loss: 2.5787 - root_mean_square
d_error: 1.6051 - val_loss: 13.0035 - val_root_mean_squared_error: 3.6060
Epoch 103/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 703us/step - loss: 2.5040 - root_mean_square
d_error: 1.5810 - val_loss: 13.0065 - val_root_mean_squared_error: 3.6065
Epoch 104/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 697us/step - loss: 2.6672 - root_mean_square
d_error: 1.6309 - val_loss: 12.5795 - val_root_mean_squared_error: 3.5468
Epoch 105/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 688us/step - loss: 2.9101 - root_mean_square
d_error: 1.7043 - val_loss: 12.5650 - val_root_mean_squared_error: 3.5447
Epoch 106/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3.5595 - root_mean_squared_
error: 1.8825 - val_loss: 12.8624 - val_root_mean_squared_error: 3.5864
Epoch 107/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 712us/step - loss: 4.3563 - root_mean_square
d_error: 2.0800 - val_loss: 12.7511 - val_root_mean_squared_error: 3.5709
Epoch 108/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 680us/step - loss: 5.1764 - root_mean_square
d_error: 2.2692 - val_loss: 16.8670 - val_root_mean_squared_error: 4.1069
Epoch 109/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 689us/step - loss: 7.6714 - root_mean_square
d_error: 2.7636 - val_loss: 13.0400 - val_root_mean_squared_error: 3.6111
Epoch 110/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 681us/step - loss: 5.8830 - root_mean_square
d_error: 2.4218 - val_loss: 12.4099 - val_root_mean_squared_error: 3.5228
Epoch 111/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 679us/step - loss: 4.4760 - root_mean_square
d_error: 2.1134 - val_loss: 13.1180 - val_root_mean_squared_error: 3.6219
Epoch 112/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 689us/step - loss: 3.9998 - root_mean_square
d_error: 1.9986 - val_loss: 13.4010 - val_root_mean_squared_error: 3.6607
Epoch 113/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 871us/step - loss: 4.2612 - root_mean_square
d_error: 2.0626 - val_loss: 13.0740 - val_root_mean_squared_error: 3.6158
Epoch 114/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 688us/step - loss: 4.4058 - root_mean_square
d_error: 2.0970 - val_loss: 12.5201 - val_root_mean_squared_error: 3.5384
Epoch 115/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 705us/step - loss: 3.7779 - root_mean_square
d_error: 1.9409 - val_loss: 13.4575 - val_root_mean_squared_error: 3.6684
Epoch 116/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 703us/step - loss: 3.3378 - root_mean_square
d_error: 1.8265 - val_loss: 13.1170 - val_root_mean_squared_error: 3.6217
Epoch 117/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 681us/step - loss: 3.1069 - root_mean_square
d_error: 1.7616 - val_loss: 13.5738 - val_root_mean_squared_error: 3.6843
Epoch 118/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 697us/step - loss: 3.0503 - root_mean_square
d_error: 1.7461 - val_loss: 12.7244 - val_root_mean_squared_error: 3.5671
Epoch 119/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 749us/step - loss: 3.0882 - root_mean_square
d_error: 1.7568 - val_loss: 12.9766 - val_root_mean_squared_error: 3.6023
Epoch 120/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 729us/step - loss: 2.8984 - root_mean_square
d_error: 1.7023 - val_loss: 13.0243 - val_root_mean_squared_error: 3.6089
```

```
Epoch 121/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 690us/step - loss: 2.8907 - root_mean_square
d_error: 1.6999 - val_loss: 13.0060 - val_root_mean_squared_error: 3.6064
Epoch 122/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 696us/step - loss: 2.6887 - root_mean_square
d_error: 1.6393 - val_loss: 13.4210 - val_root_mean_squared_error: 3.6635
Epoch 123/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 679us/step - loss: 2.5627 - root_mean_square
d_error: 1.6002 - val_loss: 13.8300 - val_root_mean_squared_error: 3.7189
Epoch 124/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 678us/step - loss: 2.5965 - root_mean_square
d_error: 1.6105 - val_loss: 13.9374 - val_root_mean_squared_error: 3.7333
Epoch 125/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 2.6672 - root_mean_squared_
error: 1.6323 - val_loss: 13.7272 - val_root_mean_squared_error: 3.7050
Epoch 126/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 723us/step - loss: 2.6344 - root_mean_square
d_error: 1.6226 - val_loss: 13.6064 - val_root_mean_squared_error: 3.6887
Epoch 127/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 696us/step - loss: 2.8194 - root_mean_square
d_error: 1.6779 - val_loss: 13.5227 - val_root_mean_squared_error: 3.6773
Epoch 128/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 676us/step - loss: 2.7802 - root_mean_square
d_error: 1.6657 - val_loss: 13.5591 - val_root_mean_squared_error: 3.6823
Epoch 129/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 673us/step - loss: 2.8724 - root_mean_square
d_error: 1.6936 - val_loss: 13.7897 - val_root_mean_squared_error: 3.7134
Epoch 130/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 685us/step - loss: 2.5988 - root_mean_square
d_error: 1.6103 - val_loss: 13.3927 - val_root_mean_squared_error: 3.6596
Epoch 131/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 678us/step - loss: 2.5568 - root_mean_square
d_error: 1.5973 - val_loss: 13.4324 - val_root_mean_squared_error: 3.6650
Epoch 132/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 938us/step - loss: 2.0144 - root_mean_square
d_error: 1.4171 - val_loss: 13.1908 - val_root_mean_squared_error: 3.6319
Epoch 133/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 714us/step - loss: 1.8614 - root_mean_square
d_error: 1.3619 - val_loss: 12.9379 - val_root_mean_squared_error: 3.5969
Epoch 134/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 671us/step - loss: 1.7084 - root_mean_square
d_error: 1.3056 - val_loss: 13.4764 - val_root_mean_squared_error: 3.6710
Epoch 135/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 690us/step - loss: 1.6990 - root_mean_square
d_error: 1.3015 - val_loss: 13.0058 - val_root_mean_squared_error: 3.6064
Epoch 136/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 685us/step - loss: 1.6455 - root_mean_square
d_error: 1.2810 - val_loss: 12.9705 - val_root_mean_squared_error: 3.6015
Epoch 137/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 1.6010 - root_mean_squared_
error: 1.2634 - val_loss: 13.1869 - val_root_mean_squared_error: 3.6314
Epoch 138/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 694us/step - loss: 1.6115 - root_mean_square
d_error: 1.2680 - val_loss: 13.1056 - val_root_mean_squared_error: 3.6202
Epoch 139/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 706us/step - loss: 1.6528 - root_mean_square
d_error: 1.2839 - val_loss: 13.0959 - val_root_mean_squared_error: 3.6188
Epoch 140/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 671us/step - loss: 1.6560 - root_mean_square
d_error: 1.2854 - val_loss: 13.2545 - val_root_mean_squared_error: 3.6407
```

```
Epoch 141/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 681us/step - loss: 1.6629 - root_mean_square
d_error: 1.2888 - val_loss: 13.3909 - val_root_mean_squared_error: 3.6594
Epoch 142/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 681us/step - loss: 1.6721 - root_mean_square
d_error: 1.2913 - val_loss: 13.7814 - val_root_mean_squared_error: 3.7123
Epoch 143/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 688us/step - loss: 1.7824 - root_mean_square
d_error: 1.3331 - val_loss: 14.2493 - val_root_mean_squared_error: 3.7748
Epoch 144/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 801us/step - loss: 1.9856 - root_mean_square
d_error: 1.4070 - val_loss: 13.8847 - val_root_mean_squared_error: 3.7262
Epoch 145/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 793us/step - loss: 1.8017 - root_mean_square
d_error: 1.3412 - val_loss: 14.6115 - val_root_mean_squared_error: 3.8225
Epoch 146/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 702us/step - loss: 2.1855 - root_mean_square
d_error: 1.4759 - val_loss: 14.6026 - val_root_mean_squared_error: 3.8213
Epoch 147/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 697us/step - loss: 2.3497 - root_mean_square
d_error: 1.5322 - val_loss: 14.5757 - val_root_mean_squared_error: 3.8178
Epoch 148/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 666us/step - loss: 2.6551 - root_mean_square
d_error: 1.6279 - val_loss: 13.7933 - val_root_mean_squared_error: 3.7139
Epoch 149/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 693us/step - loss: 2.7672 - root_mean_square
d_error: 1.6586 - val_loss: 14.1237 - val_root_mean_squared_error: 3.7582
Epoch 150/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 681us/step - loss: 2.6364 - root_mean_square
d_error: 1.6220 - val_loss: 14.5129 - val_root_mean_squared_error: 3.8096
Epoch 151/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 2.4935 - root_mean_squared_
error: 1.5782 - val_loss: 13.9048 - val_root_mean_squared_error: 3.7289
Epoch 152/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 797us/step - loss: 1.6991 - root_mean_square
d_error: 1.3025 - val_loss: 14.2809 - val_root_mean_squared_error: 3.7790
Epoch 153/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 703us/step - loss: 1.4818 - root_mean_square
d_error: 1.2161 - val_loss: 14.0216 - val_root_mean_squared_error: 3.7445
Epoch 154/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 689us/step - loss: 1.2930 - root_mean_square
d_error: 1.1359 - val_loss: 14.4956 - val_root_mean_squared_error: 3.8073
Epoch 155/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 682us/step - loss: 1.2239 - root_mean_square
d_error: 1.1049 - val_loss: 14.1010 - val_root_mean_squared_error: 3.7551
Epoch 156/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 672us/step - loss: 1.1550 - root_mean_square
d_error: 1.0739 - val_loss: 14.4255 - val_root_mean_squared_error: 3.7981
Epoch 157/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 689us/step - loss: 1.1042 - root_mean_square
d_error: 1.0497 - val_loss: 14.5217 - val_root_mean_squared_error: 3.8107
Epoch 158/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 638us/step - loss: 1.1202 - root_mean_square
d_error: 1.0575 - val_loss: 14.5517 - val_root_mean_squared_error: 3.8147
Epoch 159/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 689us/step - loss: 1.0927 - root_mean_square
d_error: 1.0445 - val_loss: 14.8714 - val_root_mean_squared_error: 3.8563
Epoch 160/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 674us/step - loss: 1.0671 - root_mean_square
d_error: 1.0324 - val_loss: 14.8491 - val_root_mean_squared_error: 3.8534
```

```
Epoch 161/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 642us/step - loss: 1.0461 - root_mean_square
d_error: 1.0222 - val_loss: 14.8374 - val_root_mean_squared_error: 3.8519
Epoch 162/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 698us/step - loss: 1.0062 - root_mean_square
d_error: 1.0027 - val_loss: 15.2961 - val_root_mean_squared_error: 3.9110
Epoch 163/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 686us/step - loss: 1.0205 - root_mean_square
d_error: 1.0098 - val_loss: 15.1510 - val_root_mean_squared_error: 3.8924
Epoch 164/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 642us/step - loss: 0.9847 - root_mean_square
d_error: 0.9921 - val_loss: 15.2639 - val_root_mean_squared_error: 3.9069
Epoch 165/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 670us/step - loss: 1.0251 - root_mean_square
d_error: 1.0121 - val_loss: 15.5720 - val_root_mean_squared_error: 3.9461
Epoch 166/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 658us/step - loss: 1.1399 - root_mean_square
d_error: 1.0673 - val_loss: 15.5653 - val_root_mean_squared_error: 3.9453
Epoch 167/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 700us/step - loss: 1.2349 - root_mean_square
d_error: 1.1095 - val_loss: 15.0270 - val_root_mean_squared_error: 3.8765
Epoch 168/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 648us/step - loss: 1.3106 - root_mean_square
d_error: 1.1415 - val_loss: 15.0529 - val_root_mean_squared_error: 3.8798
Epoch 169/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 690us/step - loss: 1.3597 - root_mean_square
d_error: 1.1619 - val_loss: 15.0503 - val_root_mean_squared_error: 3.8795
Epoch 170/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 666us/step - loss: 1.3283 - root_mean_square
d_error: 1.1498 - val_loss: 14.8757 - val_root_mean_squared_error: 3.8569
Epoch 171/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 643us/step - loss: 1.4649 - root_mean_square
d_error: 1.2071 - val_loss: 14.6648 - val_root_mean_squared_error: 3.8295
Epoch 172/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 676us/step - loss: 1.4420 - root_mean_square
d_error: 1.1975 - val_loss: 14.0554 - val_root_mean_squared_error: 3.7490
Epoch 173/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 637us/step - loss: 1.4749 - root_mean_square
d_error: 1.2099 - val_loss: 14.2042 - val_root_mean_squared_error: 3.7688
Epoch 174/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 696us/step - loss: 1.6021 - root_mean_square
d_error: 1.2625 - val_loss: 14.4059 - val_root_mean_squared_error: 3.7955
Epoch 175/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 708us/step - loss: 1.7499 - root_mean_square
d_error: 1.3194 - val_loss: 14.5995 - val_root_mean_squared_error: 3.8209
Epoch 176/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 638us/step - loss: 2.0116 - root_mean_square
d_error: 1.4163 - val_loss: 14.3369 - val_root_mean_squared_error: 3.7864
Epoch 177/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 665us/step - loss: 2.3966 - root_mean_square
d_error: 1.5468 - val_loss: 14.8228 - val_root_mean_squared_error: 3.8500
Epoch 178/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 655us/step - loss: 2.3075 - root_mean_square
d_error: 1.5182 - val_loss: 15.3083 - val_root_mean_squared_error: 3.9126
Epoch 179/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 663us/step - loss: 2.4843 - root_mean_square
d_error: 1.5746 - val_loss: 14.8492 - val_root_mean_squared_error: 3.8535
Epoch 180/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 651us/step - loss: 2.2460 - root_mean_square
d_error: 1.4983 - val_loss: 13.7394 - val_root_mean_squared_error: 3.7067
```

```
Epoch 181/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 641us/step - loss: 2.0820 - root_mean_square
d_error: 1.4422 - val_loss: 14.3164 - val_root_mean_squared_error: 3.7837
Epoch 182/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 639us/step - loss: 1.9032 - root_mean_square
d_error: 1.3794 - val_loss: 14.1163 - val_root_mean_squared_error: 3.7572
Epoch 183/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 684us/step - loss: 1.4489 - root_mean_square
d_error: 1.2032 - val_loss: 14.2218 - val_root_mean_squared_error: 3.7712
Epoch 184/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 659us/step - loss: 1.2488 - root_mean_square
d_error: 1.1168 - val_loss: 14.4404 - val_root_mean_squared_error: 3.8000
Epoch 185/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 642us/step - loss: 1.1240 - root_mean_square
d_error: 1.0597 - val_loss: 14.3950 - val_root_mean_squared_error: 3.7941
Epoch 186/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 646us/step - loss: 1.0631 - root_mean_square
d_error: 1.0303 - val_loss: 14.5454 - val_root_mean_squared_error: 3.8138
Epoch 187/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 669us/step - loss: 1.0453 - root_mean_square
d_error: 1.0217 - val_loss: 14.5794 - val_root_mean_squared_error: 3.8183
Epoch 188/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 661us/step - loss: 0.9837 - root_mean_square
d_error: 0.9913 - val_loss: 14.5357 - val_root_mean_squared_error: 3.8126
Epoch 189/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 865us/step - loss: 1.0031 - root_mean_square
d_error: 1.0012 - val_loss: 14.8898 - val_root_mean_squared_error: 3.8587
Epoch 190/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 652us/step - loss: 1.0211 - root_mean_square
d_error: 1.0101 - val_loss: 14.9016 - val_root_mean_squared_error: 3.8603
Epoch 191/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 672us/step - loss: 1.1582 - root_mean_square
d_error: 1.0751 - val_loss: 15.4860 - val_root_mean_squared_error: 3.9352
Epoch 192/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 656us/step - loss: 1.1958 - root_mean_square
d_error: 1.0918 - val_loss: 14.5832 - val_root_mean_squared_error: 3.8188
Epoch 193/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 670us/step - loss: 1.2497 - root_mean_square
d_error: 1.1165 - val_loss: 15.0186 - val_root_mean_squared_error: 3.8754
Epoch 194/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 641us/step - loss: 1.2153 - root_mean_square
d_error: 1.1014 - val_loss: 14.6379 - val_root_mean_squared_error: 3.8260
Epoch 195/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 695us/step - loss: 1.1492 - root_mean_square
d_error: 1.0713 - val_loss: 14.4368 - val_root_mean_squared_error: 3.7996
Epoch 196/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 656us/step - loss: 1.1816 - root_mean_square
d_error: 1.0864 - val_loss: 14.0256 - val_root_mean_squared_error: 3.7451
Epoch 197/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 637us/step - loss: 1.3725 - root_mean_square
d_error: 1.1695 - val_loss: 14.0634 - val_root_mean_squared_error: 3.7501
Epoch 198/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 634us/step - loss: 1.7757 - root_mean_square
d_error: 1.3293 - val_loss: 14.2638 - val_root_mean_squared_error: 3.7768
Epoch 199/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 633us/step - loss: 1.7002 - root_mean_square
d_error: 1.2928 - val_loss: 15.0092 - val_root_mean_squared_error: 3.8742
Epoch 200/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 682us/step - loss: 1.3631 - root_mean_square
d_error: 1.1628 - val_loss: 14.6831 - val_root_mean_squared_error: 3.8319
```

In [55]:
```
plot_history(history, ['root_mean_squared_error', 'val_root_mean_squared_
```

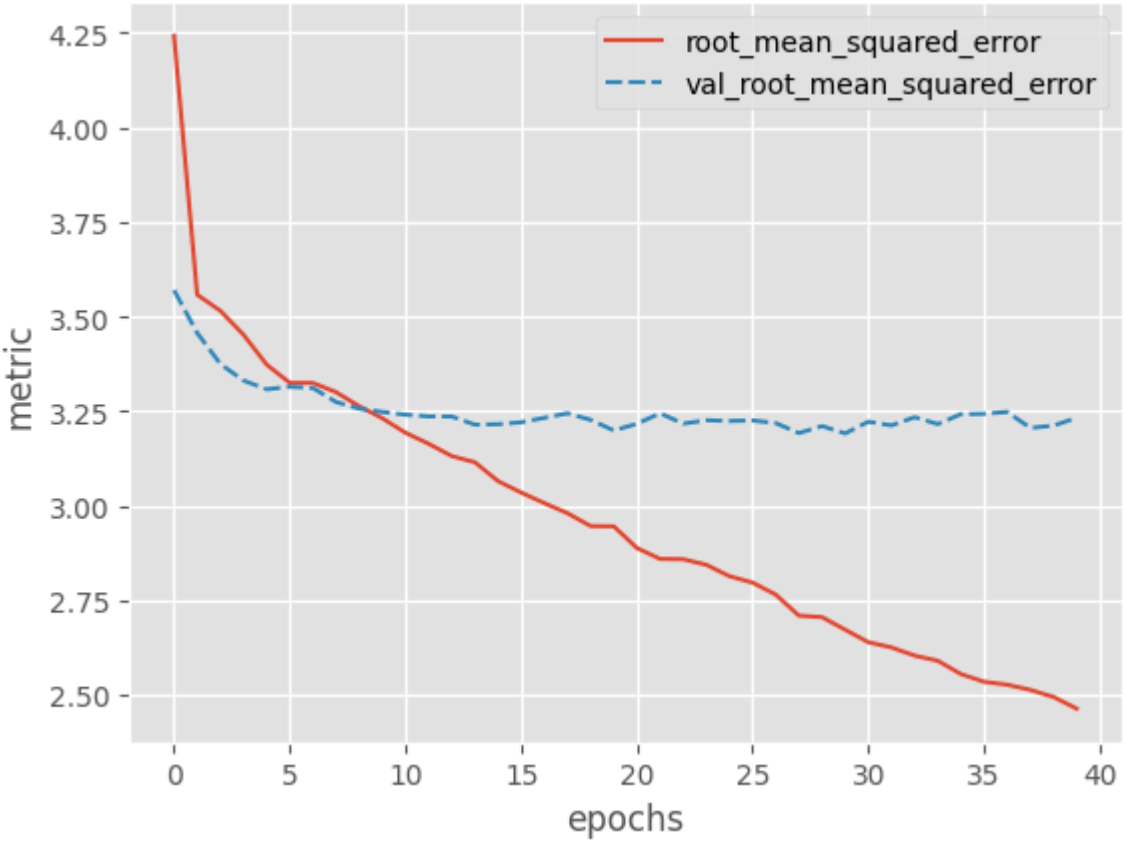
In [56]:
```
model = create_nn()
compile_model(model)
```

In [57]:
```
from tensorflow.keras.callbacks import EarlyStopping

earlystopper = EarlyStopping(
    monitor='val_loss',
    patience=10
    )

history = model.fit(X_train, y_train,
                    batch_size = 32,
                    epochs = 200,
                    validation_data=(X_val, y_val),
                    callbacks=[earlystopper])
```

```
Epoch 1/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 21.9496 - root_mean_squared
_error: 4.6639 - val_loss: 12.7446 - val_root_mean_squared_error: 3.5700
Epoch 2/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 612us/step - loss: 13.6725 - root_mean_squar
ed_error: 3.6939 - val_loss: 11.9496 - val_root_mean_squared_error: 3.4568
Epoch 3/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 653us/step - loss: 13.1376 - root_mean_squar
ed_error: 3.6230 - val_loss: 11.3892 - val_root_mean_squared_error: 3.3748
Epoch 4/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 698us/step - loss: 12.7822 - root_mean_squar
ed_error: 3.5734 - val_loss: 11.0952 - val_root_mean_squared_error: 3.3310
Epoch 5/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 622us/step - loss: 12.2842 - root_mean_squar
ed_error: 3.5025 - val_loss: 10.9414 - val_root_mean_squared_error: 3.3078
Epoch 6/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 681us/step - loss: 11.9116 - root_mean_squar
ed_error: 3.4488 - val_loss: 10.9865 - val_root_mean_squared_error: 3.3146
Epoch 7/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 676us/step - loss: 11.8474 - root_mean_squar
ed_error: 3.4401 - val_loss: 10.9635 - val_root_mean_squared_error: 3.3111
Epoch 8/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 684us/step - loss: 11.6855 - root_mean_squar
ed_error: 3.4167 - val_loss: 10.7201 - val_root_mean_squared_error: 3.2742
Epoch 9/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 667us/step - loss: 11.4949 - root_mean_squar
ed_error: 3.3882 - val_loss: 10.6048 - val_root_mean_squared_error: 3.2565
Epoch 10/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 661us/step - loss: 11.2531 - root_mean_squar
ed_error: 3.3524 - val_loss: 10.5483 - val_root_mean_squared_error: 3.2478
Epoch 11/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 679us/step - loss: 11.0153 - root_mean_squar
ed_error: 3.3166 - val_loss: 10.5014 - val_root_mean_squared_error: 3.2406
Epoch 12/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 689us/step - loss: 10.8393 - root_mean_squar
ed_error: 3.2898 - val_loss: 10.4722 - val_root_mean_squared_error: 3.2361
Epoch 13/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 677us/step - loss: 10.6026 - root_mean_squar
ed_error: 3.2538 - val_loss: 10.4702 - val_root_mean_squared_error: 3.2358
Epoch 14/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 673us/step - loss: 10.5261 - root_mean_squar
ed_error: 3.2420 - val_loss: 10.3312 - val_root_mean_squared_error: 3.2142
Epoch 15/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 675us/step - loss: 10.1652 - root_mean_squar
ed_error: 3.1860 - val_loss: 10.3400 - val_root_mean_squared_error: 3.2156
Epoch 16/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 679us/step - loss: 9.9794 - root_mean_square
d_error: 3.1565 - val_loss: 10.3719 - val_root_mean_squared_error: 3.2205
Epoch 17/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 674us/step - loss: 9.7146 - root_mean_square
d_error: 3.1147 - val_loss: 10.4475 - val_root_mean_squared_error: 3.2323
Epoch 18/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 674us/step - loss: 9.6135 - root_mean_square
d_error: 3.0984 - val_loss: 10.5257 - val_root_mean_squared_error: 3.2443
Epoch 19/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 660us/step - loss: 9.3583 - root_mean_square
d_error: 3.0571 - val_loss: 10.4122 - val_root_mean_squared_error: 3.2268
Epoch 20/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 677us/step - loss: 9.3313 - root_mean_square
d_error: 3.0529 - val_loss: 10.2329 - val_root_mean_squared_error: 3.1989
```

```
Epoch 21/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 684us/step - loss: 8.9913 - root_mean_square
d_error: 2.9966 - val_loss: 10.3460 - val_root_mean_squared_error: 3.2165
Epoch 22/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 653us/step - loss: 8.7345 - root_mean_square
d_error: 2.9538 - val_loss: 10.5252 - val_root_mean_squared_error: 3.2442
Epoch 23/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 661us/step - loss: 8.7594 - root_mean_square
d_error: 2.9580 - val_loss: 10.3484 - val_root_mean_squared_error: 3.2169
Epoch 24/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 678us/step - loss: 8.6321 - root_mean_square
d_error: 2.9366 - val_loss: 10.4071 - val_root_mean_squared_error: 3.2260
Epoch 25/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 667us/step - loss: 8.5102 - root_mean_square
d_error: 2.9156 - val_loss: 10.3939 - val_root_mean_squared_error: 3.2240
Epoch 26/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 681us/step - loss: 8.3799 - root_mean_square
d_error: 2.8933 - val_loss: 10.4024 - val_root_mean_squared_error: 3.2253
Epoch 27/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 663us/step - loss: 8.1587 - root_mean_square
d_error: 2.8550 - val_loss: 10.3582 - val_root_mean_squared_error: 3.2184
Epoch 28/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 663us/step - loss: 7.9089 - root_mean_square
d_error: 2.8106 - val_loss: 10.1888 - val_root_mean_squared_error: 3.1920
Epoch 29/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 660us/step - loss: 7.8282 - root_mean_square
d_error: 2.7963 - val_loss: 10.3069 - val_root_mean_squared_error: 3.2104
Epoch 30/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 653us/step - loss: 7.6261 - root_mean_square
d_error: 2.7601 - val_loss: 10.1829 - val_root_mean_squared_error: 3.1911
Epoch 31/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 660us/step - loss: 7.4872 - root_mean_square
d_error: 2.7345 - val_loss: 10.3813 - val_root_mean_squared_error: 3.2220
Epoch 32/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 666us/step - loss: 7.4355 - root_mean_square
d_error: 2.7249 - val_loss: 10.3188 - val_root_mean_squared_error: 3.2123
Epoch 33/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 675us/step - loss: 7.2872 - root_mean_square
d_error: 2.6977 - val_loss: 10.4587 - val_root_mean_squared_error: 3.2340
Epoch 34/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 668us/step - loss: 7.2111 - root_mean_square
d_error: 2.6839 - val_loss: 10.3406 - val_root_mean_squared_error: 3.2157
Epoch 35/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 659us/step - loss: 7.0539 - root_mean_square
d_error: 2.6543 - val_loss: 10.5105 - val_root_mean_squared_error: 3.2420
Epoch 36/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 661us/step - loss: 6.9004 - root_mean_square
d_error: 2.6252 - val_loss: 10.5167 - val_root_mean_squared_error: 3.2429
Epoch 37/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 682us/step - loss: 6.8772 - root_mean_square
d_error: 2.6208 - val_loss: 10.5483 - val_root_mean_squared_error: 3.2478
Epoch 38/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 677us/step - loss: 6.7726 - root_mean_square
d_error: 2.6011 - val_loss: 10.2760 - val_root_mean_squared_error: 3.2056
Epoch 39/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 688us/step - loss: 6.7528 - root_mean_square
d_error: 2.5966 - val_loss: 10.3153 - val_root_mean_squared_error: 3.2118
Epoch 40/200
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 669us/step - loss: 6.5755 - root_mean_square
d_error: 2.5624 - val_loss: 10.4482 - val_root_mean_squared_error: 3.2324
```

In [58]:
```
plot_history(history, ['root_mean_squared_error', 'val_root_mean_squared_
```



In [59]:
```
def create_nn():
    # Input layer
    inputs = keras.layers.Input(shape=(X_data.shape[1],), name='input')

    # Dense layers
    layers_dense = keras.layers.BatchNormalization()(inputs) # This is ne
    layers_dense = keras.layers.Dense(100, 'relu')(layers_dense)
    layers_dense = keras.layers.Dense(50, 'relu')(layers_dense)

    # Output layer
    outputs = keras.layers.Dense(1)(layers_dense)

    # Defining the model and compiling it
    return keras.Model(inputs=inputs, outputs=outputs, name="model_batchn

model = create_nn()
compile_model(model)
model.summary()
```

Model: "model_batchnorm"

Layer (type)	Output Shape	
input (InputLayer)	(None, 89)	
batch_normalization (BatchNormalization)	(None, 89)	
dense_11 (Dense)	(None, 100)	
dense_12 (Dense)	(None, 50)	
dense_13 (Dense)	(None, 1)	

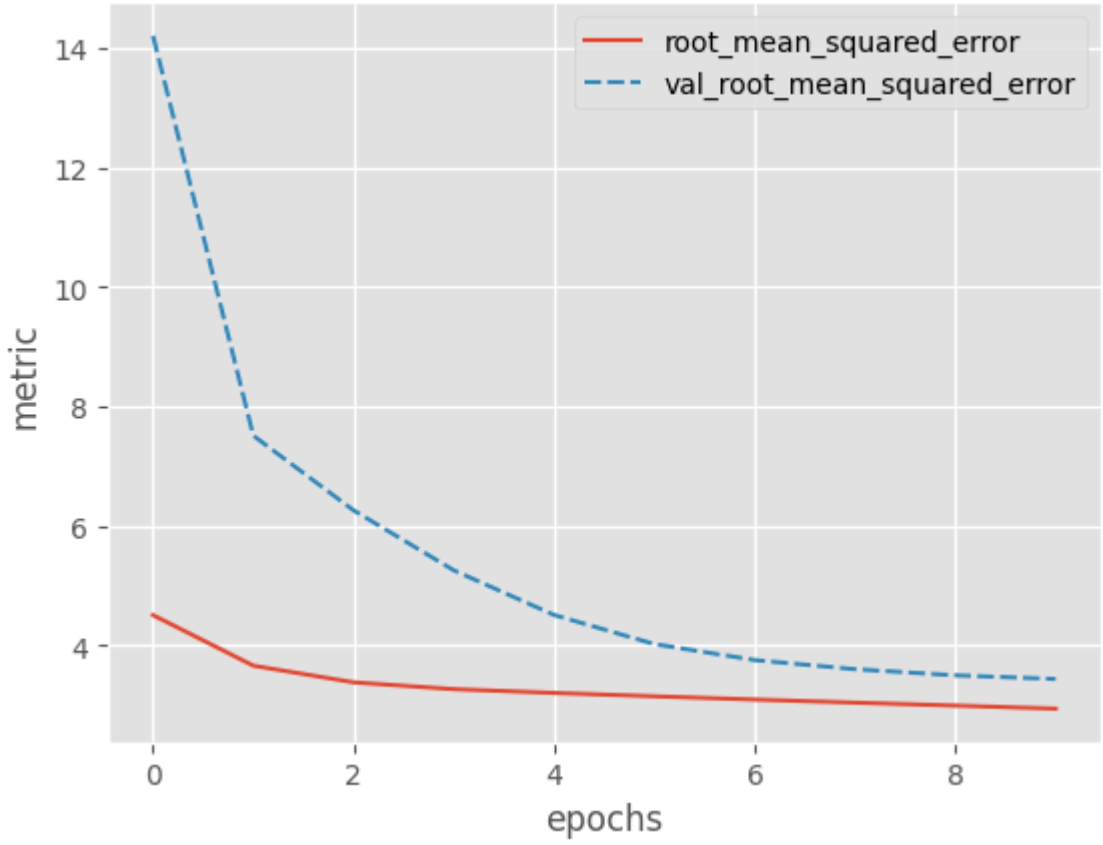
Total params: 14,457 (56.47 KB)

Trainable params: 14,279 (55.78 KB)

Non-trainable params: 178 (712.00 B)

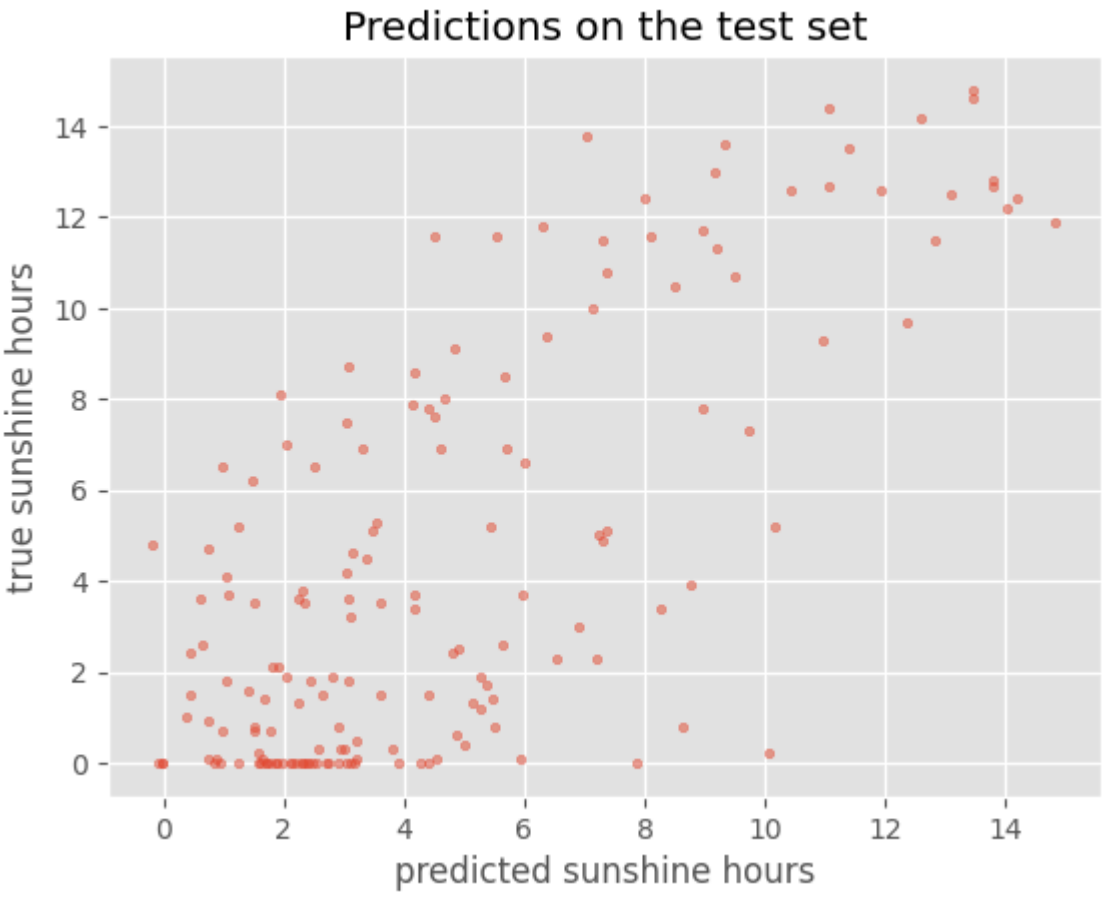
In [60]:
```
history = model.fit(X_train, y_train,
                    batch_size = 32,
                    epochs = 1000,
                    validation_data=(X_val, y_val),
                    callbacks=[earlystopper])

plot_history(history, ['root_mean_squared_error', 'val_root_mean_squared_
```

```
Epoch 1/1000
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 26.8749 - root_mean_squared
_error: 5.1405 - val_loss: 201.5687 - val_root_mean_squared_error: 14.1975
Epoch 2/1000
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 708us/step - loss: 14.5041 - root_mean_squar
ed_error: 3.8036 - val_loss: 56.3645 - val_root_mean_squared_error: 7.5076
Epoch 3/1000
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 794us/step - loss: 12.2765 - root_mean_squar
ed_error: 3.4977 - val_loss: 39.1516 - val_root_mean_squared_error: 6.2571
Epoch 4/1000
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 782us/step - loss: 11.4693 - root_mean_squar
ed_error: 3.3803 - val_loss: 27.5959 - val_root_mean_squared_error: 5.2532
Epoch 5/1000
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 729us/step - loss: 11.0782 - root_mean_squar
ed_error: 3.3220 - val_loss: 20.3016 - val_root_mean_squared_error: 4.5057
Epoch 6/1000
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 10.7333 - root_mean_squared
_error: 3.2698 - val_loss: 16.1555 - val_root_mean_squared_error: 4.0194
Epoch 7/1000
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 742us/step - loss: 10.3788 - root_mean_squar
ed_error: 3.2152 - val_loss: 14.0763 - val_root_mean_squared_error: 3.7518
Epoch 8/1000
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 701us/step - loss: 10.0256 - root_mean_squar
ed_error: 3.1601 - val_loss: 12.9569 - val_root_mean_squared_error: 3.5996
Epoch 9/1000
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 712us/step - loss: 9.6927 - root_mean_square
d_error: 3.1075 - val_loss: 12.2268 - val_root_mean_squared_error: 3.4967
Epoch 10/1000
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 725us/step - loss: 9.3515 - root_mean_square
d_error: 3.0526 - val_loss: 11.8062 - val_root_mean_squared_error: 3.4360
```



In [61]:
```
y_test_predicted = model.predict(X_test)
plot_predictions(y_test_predicted, y_test, title='Predictions on the test
```

```
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step
```



In [62]:
```
nr_rows = 365*9
# data
X_data = data.loc[:nr_rows].drop(columns=['DATE', 'MONTH'])

# labels (sunshine hours the next day)
y_data = data.loc[1:(nr_rows + 1)]["BASEL_sunshine"]
```

In [63]:
```
# only use columns with 'BASEL'
cols = [c for c in X_data.columns if c[:5] == 'BASEL']
X_data = X_data[cols]
```

In [64]:
```
X_train, X_holdout, y_train, y_holdout = train_test_split(X_data, y_data,
X_val, X_test, y_val, y_test = train_test_split(X_holdout, y_holdout, tes
```

In [65]:
```
# create the network and view its summary
model = create_nn()
compile_model(model)
model.summary()
```

Model: "model_batchnorm"

Layer (type)	Output Shape	P
input (InputLayer)	(None, 9)	
batch_normalization_1 (BatchNormalization)	(None, 9)	
dense_14 (Dense)	(None, 100)	
dense_15 (Dense)	(None, 50)	
dense_16 (Dense)	(None, 1)	

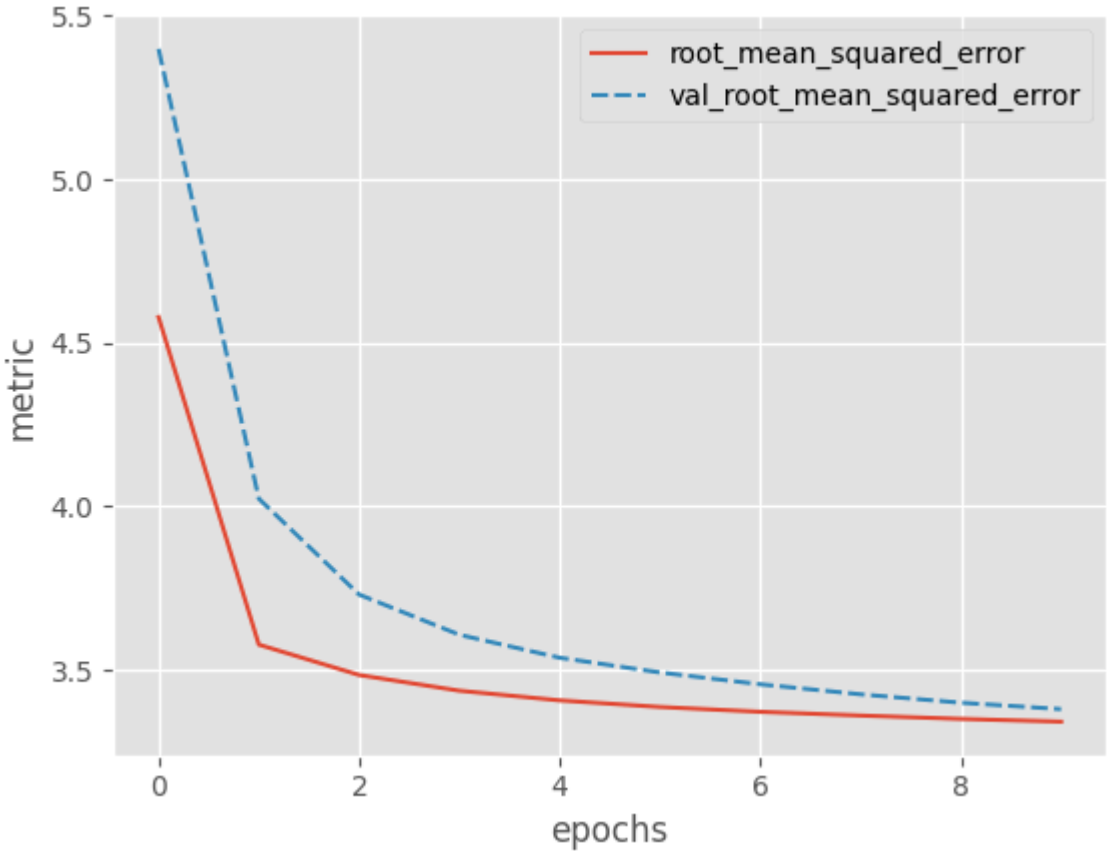
Total params: 6,137 (23.97 KB)

Trainable params: 6,119 (23.90 KB)

Non-trainable params: 18 (72.00 B)

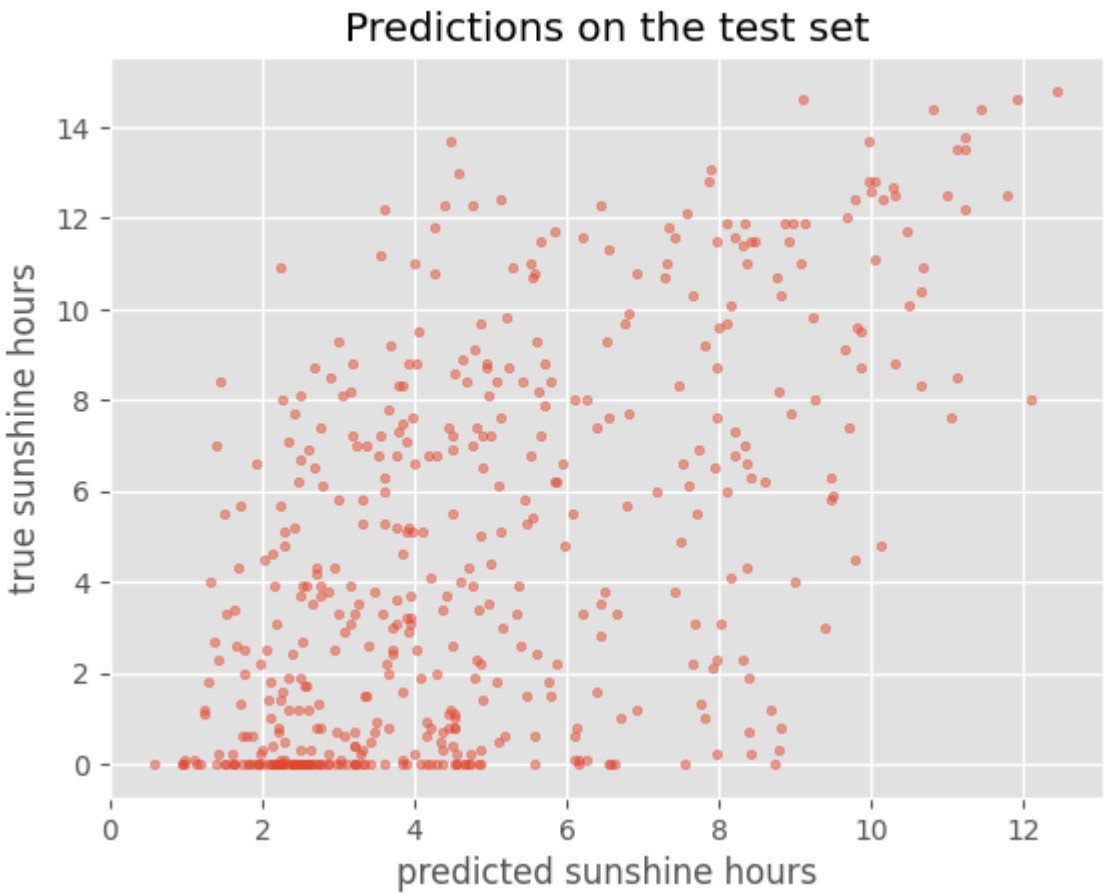
In [66]:
```
history = model.fit(X_train, y_train,
                    batch_size = 32,
                    epochs = 1000,
                    validation_data=(X_val, y_val),
                    callbacks=[earlystopper],
                    verbose = 2)
plot_history(history, ['root_mean_squared_error', 'val_root_mean_squared_
```

```
Epoch 1/1000
72/72 - 0s - 5ms/step - loss: 20.9696 - root_mean_squared_error: 4.5793 -
val_loss: 29.1530 - val_root_mean_squared_error: 5.3993
Epoch 2/1000
72/72 - 0s - 497us/step - loss: 12.7991 - root_mean_squared_error: 3.5776
- val_loss: 16.1918 - val_root_mean_squared_error: 4.0239
Epoch 3/1000
72/72 - 0s - 491us/step - loss: 12.1431 - root_mean_squared_error: 3.4847
- val_loss: 13.9184 - val_root_mean_squared_error: 3.7307
Epoch 4/1000
72/72 - 0s - 483us/step - loss: 11.8105 - root_mean_squared_error: 3.4366
- val_loss: 13.0157 - val_root_mean_squared_error: 3.6077
Epoch 5/1000
72/72 - 0s - 471us/step - loss: 11.6080 - root_mean_squared_error: 3.4070
- val_loss: 12.5137 - val_root_mean_squared_error: 3.5375
Epoch 6/1000
72/72 - 0s - 500us/step - loss: 11.4712 - root_mean_squared_error: 3.3869
- val_loss: 12.1968 - val_root_mean_squared_error: 3.4924
Epoch 7/1000
72/72 - 0s - 526us/step - loss: 11.3716 - root_mean_squared_error: 3.3722
- val_loss: 11.9460 - val_root_mean_squared_error: 3.4563
Epoch 8/1000
72/72 - 0s - 494us/step - loss: 11.2947 - root_mean_squared_error: 3.3608
- val_loss: 11.7351 - val_root_mean_squared_error: 3.4257
Epoch 9/1000
72/72 - 0s - 494us/step - loss: 11.2258 - root_mean_squared_error: 3.3505
- val_loss: 11.5600 - val_root_mean_squared_error: 3.4000
Epoch 10/1000
72/72 - 0s - 511us/step - loss: 11.1692 - root_mean_squared_error: 3.3420
- val_loss: 11.4264 - val_root_mean_squared_error: 3.3803
```



In [67]:
```
y_test_predicted = model.predict(X_test)
plot_predictions(y_test_predicted, y_test, title='Predictions on the test
```

```
16/16 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step
```



In [68]:
```
test_metrics = model.evaluate(X_test, y_test, return_dict=True)
print(f'Test RMSE: {test_metrics["root_mean_squared_error"]}')
```

```
16/16 ━━━━━━━━━━━━━━━━━━━━ 0s 419us/step - loss: 12.2155 - root_mean_squar
ed_error: 3.4919
Test RMSE: 3.413457155227661
```

In [69]:
```
from tensorflow.keras.callbacks import TensorBoard
import datetime
log_dir = "logs/fit/" + datetime.datetime.now().strftime("%Y%m%d-%H%M%S")
tensorboard_callback = TensorBoard(log_dir=log_dir, histogram_freq=1)
history = model.fit(X_train, y_train,
                    batch_size = 32,
                    epochs = 200,
                    validation_data=(X_val, y_val),
                    callbacks=[tensorboard_callback],
                    verbose = 2)
```

```
Epoch 1/200
72/72 - 0s - 2ms/step - loss: 11.1271 - root_mean_squared_error: 3.3357 -
val_loss: 11.3320 - val_root_mean_squared_error: 3.3663
Epoch 2/200
72/72 - 0s - 796us/step - loss: 11.0815 - root_mean_squared_error: 3.3289
- val_loss: 11.2867 - val_root_mean_squared_error: 3.3596
Epoch 3/200
72/72 - 0s - 797us/step - loss: 11.0481 - root_mean_squared_error: 3.3239
- val_loss: 11.2474 - val_root_mean_squared_error: 3.3537
Epoch 4/200
72/72 - 0s - 787us/step - loss: 11.0094 - root_mean_squared_error: 3.3180
- val_loss: 11.2310 - val_root_mean_squared_error: 3.3513
Epoch 5/200
72/72 - 0s - 790us/step - loss: 10.9796 - root_mean_squared_error: 3.3136
- val_loss: 11.2050 - val_root_mean_squared_error: 3.3474
Epoch 6/200
72/72 - 0s - 794us/step - loss: 10.9459 - root_mean_squared_error: 3.3085
- val_loss: 11.2049 - val_root_mean_squared_error: 3.3474
Epoch 7/200
72/72 - 0s - 794us/step - loss: 10.9223 - root_mean_squared_error: 3.3049
- val_loss: 11.1890 - val_root_mean_squared_error: 3.3450
Epoch 8/200
72/72 - 0s - 781us/step - loss: 10.8952 - root_mean_squared_error: 3.3008
- val_loss: 11.1682 - val_root_mean_squared_error: 3.3419
Epoch 9/200
72/72 - 0s - 759us/step - loss: 10.8700 - root_mean_squared_error: 3.2970
- val_loss: 11.1551 - val_root_mean_squared_error: 3.3399
Epoch 10/200
72/72 - 0s - 763us/step - loss: 10.8449 - root_mean_squared_error: 3.2932
- val_loss: 11.1498 - val_root_mean_squared_error: 3.3391
Epoch 11/200
72/72 - 0s - 1ms/step - loss: 10.8196 - root_mean_squared_error: 3.2893 -
val_loss: 11.1370 - val_root_mean_squared_error: 3.3372
Epoch 12/200
72/72 - 0s - 790us/step - loss: 10.7974 - root_mean_squared_error: 3.2859
- val_loss: 11.1306 - val_root_mean_squared_error: 3.3363
Epoch 13/200
72/72 - 0s - 804us/step - loss: 10.7766 - root_mean_squared_error: 3.2828
- val_loss: 11.1216 - val_root_mean_squared_error: 3.3349
Epoch 14/200
72/72 - 0s - 786us/step - loss: 10.7532 - root_mean_squared_error: 3.2792
- val_loss: 11.1100 - val_root_mean_squared_error: 3.3332
Epoch 15/200
72/72 - 0s - 788us/step - loss: 10.7312 - root_mean_squared_error: 3.2758
- val_loss: 11.1103 - val_root_mean_squared_error: 3.3332
Epoch 16/200
72/72 - 0s - 796us/step - loss: 10.7127 - root_mean_squared_error: 3.2730
- val_loss: 11.1050 - val_root_mean_squared_error: 3.3324
Epoch 17/200
72/72 - 0s - 786us/step - loss: 10.6953 - root_mean_squared_error: 3.2704
- val_loss: 11.0997 - val_root_mean_squared_error: 3.3316
Epoch 18/200
72/72 - 0s - 784us/step - loss: 10.6757 - root_mean_squared_error: 3.2674
- val_loss: 11.0964 - val_root_mean_squared_error: 3.3311
Epoch 19/200
72/72 - 0s - 783us/step - loss: 10.6571 - root_mean_squared_error: 3.2645
- val_loss: 11.0928 - val_root_mean_squared_error: 3.3306
Epoch 20/200
72/72 - 0s - 771us/step - loss: 10.6388 - root_mean_squared_error: 3.2617
- val_loss: 11.0964 - val_root_mean_squared_error: 3.3311
```

```
Epoch 21/200
72/72 - 0s - 786us/step - loss: 10.6196 - root_mean_squared_error: 3.2588
- val_loss: 11.0941 - val_root_mean_squared_error: 3.3308
Epoch 22/200
72/72 - 0s - 791us/step - loss: 10.6000 - root_mean_squared_error: 3.2558
- val_loss: 11.0942 - val_root_mean_squared_error: 3.3308
Epoch 23/200
72/72 - 0s - 779us/step - loss: 10.5803 - root_mean_squared_error: 3.2527
- val_loss: 11.0942 - val_root_mean_squared_error: 3.3308
Epoch 24/200
72/72 - 0s - 773us/step - loss: 10.5608 - root_mean_squared_error: 3.2497
- val_loss: 11.0960 - val_root_mean_squared_error: 3.3311
Epoch 25/200
72/72 - 0s - 787us/step - loss: 10.5437 - root_mean_squared_error: 3.2471
- val_loss: 11.0898 - val_root_mean_squared_error: 3.3301
Epoch 26/200
72/72 - 0s - 789us/step - loss: 10.5224 - root_mean_squared_error: 3.2438
- val_loss: 11.0924 - val_root_mean_squared_error: 3.3305
Epoch 27/200
72/72 - 0s - 797us/step - loss: 10.5067 - root_mean_squared_error: 3.2414
- val_loss: 11.0864 - val_root_mean_squared_error: 3.3296
Epoch 28/200
72/72 - 0s - 784us/step - loss: 10.4865 - root_mean_squared_error: 3.2383
- val_loss: 11.0869 - val_root_mean_squared_error: 3.3297
Epoch 29/200
72/72 - 0s - 793us/step - loss: 10.4772 - root_mean_squared_error: 3.2369
- val_loss: 11.0833 - val_root_mean_squared_error: 3.3292
Epoch 30/200
72/72 - 0s - 842us/step - loss: 10.4514 - root_mean_squared_error: 3.2329
- val_loss: 11.0918 - val_root_mean_squared_error: 3.3304
Epoch 31/200
72/72 - 0s - 823us/step - loss: 10.4367 - root_mean_squared_error: 3.2306
- val_loss: 11.0935 - val_root_mean_squared_error: 3.3307
Epoch 32/200
72/72 - 0s - 825us/step - loss: 10.4197 - root_mean_squared_error: 3.2280
- val_loss: 11.0943 - val_root_mean_squared_error: 3.3308
Epoch 33/200
72/72 - 0s - 801us/step - loss: 10.4016 - root_mean_squared_error: 3.2251
- val_loss: 11.1095 - val_root_mean_squared_error: 3.3331
Epoch 34/200
72/72 - 0s - 809us/step - loss: 10.3817 - root_mean_squared_error: 3.2221
- val_loss: 11.1077 - val_root_mean_squared_error: 3.3328
Epoch 35/200
72/72 - 0s - 792us/step - loss: 10.3669 - root_mean_squared_error: 3.2198
- val_loss: 11.1048 - val_root_mean_squared_error: 3.3324
Epoch 36/200
72/72 - 0s - 801us/step - loss: 10.3492 - root_mean_squared_error: 3.2170
- val_loss: 11.1126 - val_root_mean_squared_error: 3.3336
Epoch 37/200
72/72 - 0s - 793us/step - loss: 10.3288 - root_mean_squared_error: 3.2138
- val_loss: 11.1122 - val_root_mean_squared_error: 3.3335
Epoch 38/200
72/72 - 0s - 778us/step - loss: 10.3135 - root_mean_squared_error: 3.2115
- val_loss: 11.1240 - val_root_mean_squared_error: 3.3353
Epoch 39/200
72/72 - 0s - 1ms/step - loss: 10.2955 - root_mean_squared_error: 3.2087 -
val_loss: 11.1273 - val_root_mean_squared_error: 3.3358
Epoch 40/200
72/72 - 0s - 796us/step - loss: 10.2799 - root_mean_squared_error: 3.2062
- val_loss: 11.1446 - val_root_mean_squared_error: 3.3383
```

```
Epoch 41/200
72/72 - 0s - 785us/step - loss: 10.2684 - root_mean_squared_error: 3.2044
- val_loss: 11.1440 - val_root_mean_squared_error: 3.3383
Epoch 42/200
72/72 - 0s - 802us/step - loss: 10.2465 - root_mean_squared_error: 3.2010
- val_loss: 11.1610 - val_root_mean_squared_error: 3.3408
Epoch 43/200
72/72 - 0s - 805us/step - loss: 10.2312 - root_mean_squared_error: 3.1986
- val_loss: 11.1669 - val_root_mean_squared_error: 3.3417
Epoch 44/200
72/72 - 0s - 813us/step - loss: 10.2127 - root_mean_squared_error: 3.1957
- val_loss: 11.1684 - val_root_mean_squared_error: 3.3419
Epoch 45/200
72/72 - 0s - 783us/step - loss: 10.1963 - root_mean_squared_error: 3.1932
- val_loss: 11.1798 - val_root_mean_squared_error: 3.3436
Epoch 46/200
72/72 - 0s - 799us/step - loss: 10.1843 - root_mean_squared_error: 3.1913
- val_loss: 11.1886 - val_root_mean_squared_error: 3.3449
Epoch 47/200
72/72 - 0s - 806us/step - loss: 10.1659 - root_mean_squared_error: 3.1884
- val_loss: 11.1796 - val_root_mean_squared_error: 3.3436
Epoch 48/200
72/72 - 0s - 800us/step - loss: 10.1471 - root_mean_squared_error: 3.1855
- val_loss: 11.1967 - val_root_mean_squared_error: 3.3461
Epoch 49/200
72/72 - 0s - 806us/step - loss: 10.1273 - root_mean_squared_error: 3.1823
- val_loss: 11.2082 - val_root_mean_squared_error: 3.3479
Epoch 50/200
72/72 - 0s - 807us/step - loss: 10.1190 - root_mean_squared_error: 3.1810
- val_loss: 11.2000 - val_root_mean_squared_error: 3.3466
Epoch 51/200
72/72 - 0s - 799us/step - loss: 10.0958 - root_mean_squared_error: 3.1774
- val_loss: 11.2054 - val_root_mean_squared_error: 3.3475
Epoch 52/200
72/72 - 0s - 780us/step - loss: 10.0862 - root_mean_squared_error: 3.1759
- val_loss: 11.2237 - val_root_mean_squared_error: 3.3502
Epoch 53/200
72/72 - 0s - 792us/step - loss: 10.0628 - root_mean_squared_error: 3.1722
- val_loss: 11.2306 - val_root_mean_squared_error: 3.3512
Epoch 54/200
72/72 - 0s - 793us/step - loss: 10.0474 - root_mean_squared_error: 3.1698
- val_loss: 11.2515 - val_root_mean_squared_error: 3.3543
Epoch 55/200
72/72 - 0s - 789us/step - loss: 10.0306 - root_mean_squared_error: 3.1671
- val_loss: 11.2584 - val_root_mean_squared_error: 3.3554
Epoch 56/200
72/72 - 0s - 792us/step - loss: 10.0138 - root_mean_squared_error: 3.1645
- val_loss: 11.2465 - val_root_mean_squared_error: 3.3536
Epoch 57/200
72/72 - 0s - 797us/step - loss: 9.9957 - root_mean_squared_error: 3.1616 -
val_loss: 11.2870 - val_root_mean_squared_error: 3.3596
Epoch 58/200
72/72 - 0s - 799us/step - loss: 9.9867 - root_mean_squared_error: 3.1602 -
val_loss: 11.2868 - val_root_mean_squared_error: 3.3596
Epoch 59/200
72/72 - 0s - 800us/step - loss: 9.9682 - root_mean_squared_error: 3.1572 -
val_loss: 11.2876 - val_root_mean_squared_error: 3.3597
Epoch 60/200
72/72 - 0s - 810us/step - loss: 9.9472 - root_mean_squared_error: 3.1539 -
val_loss: 11.3046 - val_root_mean_squared_error: 3.3622
```

```
Epoch 61/200
72/72 - 0s - 787us/step - loss: 9.9305 - root_mean_squared_error: 3.1513 -
val_loss: 11.3171 - val_root_mean_squared_error: 3.3641
Epoch 62/200
72/72 - 0s - 799us/step - loss: 9.9112 - root_mean_squared_error: 3.1482 -
val_loss: 11.3306 - val_root_mean_squared_error: 3.3661
Epoch 63/200
72/72 - 0s - 792us/step - loss: 9.9002 - root_mean_squared_error: 3.1465 -
val_loss: 11.3427 - val_root_mean_squared_error: 3.3679
Epoch 64/200
72/72 - 0s - 807us/step - loss: 9.8808 - root_mean_squared_error: 3.1434 -
val_loss: 11.3481 - val_root_mean_squared_error: 3.3687
Epoch 65/200
72/72 - 0s - 797us/step - loss: 9.8679 - root_mean_squared_error: 3.1413 -
val_loss: 11.3594 - val_root_mean_squared_error: 3.3704
Epoch 66/200
72/72 - 0s - 810us/step - loss: 9.8545 - root_mean_squared_error: 3.1392 -
val_loss: 11.3617 - val_root_mean_squared_error: 3.3707
Epoch 67/200
72/72 - 0s - 793us/step - loss: 9.8286 - root_mean_squared_error: 3.1351 -
val_loss: 11.3790 - val_root_mean_squared_error: 3.3733
Epoch 68/200
72/72 - 0s - 784us/step - loss: 9.8169 - root_mean_squared_error: 3.1332 -
val_loss: 11.3794 - val_root_mean_squared_error: 3.3733
Epoch 69/200
72/72 - 0s - 798us/step - loss: 9.7986 - root_mean_squared_error: 3.1303 -
val_loss: 11.4069 - val_root_mean_squared_error: 3.3774
Epoch 70/200
72/72 - 0s - 802us/step - loss: 9.7783 - root_mean_squared_error: 3.1270 -
val_loss: 11.4183 - val_root_mean_squared_error: 3.3791
Epoch 71/200
72/72 - 0s - 801us/step - loss: 9.7641 - root_mean_squared_error: 3.1248 -
val_loss: 11.4540 - val_root_mean_squared_error: 3.3844
Epoch 72/200
72/72 - 0s - 802us/step - loss: 9.7518 - root_mean_squared_error: 3.1228 -
val_loss: 11.4457 - val_root_mean_squared_error: 3.3832
Epoch 73/200
72/72 - 0s - 788us/step - loss: 9.7223 - root_mean_squared_error: 3.1181 -
val_loss: 11.4375 - val_root_mean_squared_error: 3.3819
Epoch 74/200
72/72 - 0s - 778us/step - loss: 9.7097 - root_mean_squared_error: 3.1160 -
val_loss: 11.4594 - val_root_mean_squared_error: 3.3852
Epoch 75/200
72/72 - 0s - 783us/step - loss: 9.6952 - root_mean_squared_error: 3.1137 -
val_loss: 11.4642 - val_root_mean_squared_error: 3.3859
Epoch 76/200
72/72 - 0s - 800us/step - loss: 9.6684 - root_mean_squared_error: 3.1094 -
val_loss: 11.4660 - val_root_mean_squared_error: 3.3861
Epoch 77/200
72/72 - 0s - 809us/step - loss: 9.6551 - root_mean_squared_error: 3.1073 -
val_loss: 11.5341 - val_root_mean_squared_error: 3.3962
Epoch 78/200
72/72 - 0s - 796us/step - loss: 9.6410 - root_mean_squared_error: 3.1050 -
val_loss: 11.4773 - val_root_mean_squared_error: 3.3878
Epoch 79/200
72/72 - 0s - 797us/step - loss: 9.6240 - root_mean_squared_error: 3.1022 -
val_loss: 11.5268 - val_root_mean_squared_error: 3.3951
Epoch 80/200
72/72 - 0s - 792us/step - loss: 9.5996 - root_mean_squared_error: 3.0983 -
val_loss: 11.5094 - val_root_mean_squared_error: 3.3925
```

```
Epoch 81/200
72/72 - 0s - 799us/step - loss: 9.5845 - root_mean_squared_error: 3.0959 -
val_loss: 11.5556 - val_root_mean_squared_error: 3.3994
Epoch 82/200
72/72 - 0s - 796us/step - loss: 9.5665 - root_mean_squared_error: 3.0930 -
val_loss: 11.5682 - val_root_mean_squared_error: 3.4012
Epoch 83/200
72/72 - 0s - 791us/step - loss: 9.5519 - root_mean_squared_error: 3.0906 -
val_loss: 11.5485 - val_root_mean_squared_error: 3.3983
Epoch 84/200
72/72 - 0s - 1ms/step - loss: 9.5263 - root_mean_squared_error: 3.0865 - v
al_loss: 11.5573 - val_root_mean_squared_error: 3.3996
Epoch 85/200
72/72 - 0s - 823us/step - loss: 9.5126 - root_mean_squared_error: 3.0843 -
val_loss: 11.5987 - val_root_mean_squared_error: 3.4057
Epoch 86/200
72/72 - 0s - 795us/step - loss: 9.4916 - root_mean_squared_error: 3.0808 -
val_loss: 11.5846 - val_root_mean_squared_error: 3.4036
Epoch 87/200
72/72 - 0s - 806us/step - loss: 9.4825 - root_mean_squared_error: 3.0794 -
val_loss: 11.6083 - val_root_mean_squared_error: 3.4071
Epoch 88/200
72/72 - 0s - 811us/step - loss: 9.4700 - root_mean_squared_error: 3.0773 -
val_loss: 11.5937 - val_root_mean_squared_error: 3.4050
Epoch 89/200
72/72 - 0s - 812us/step - loss: 9.4517 - root_mean_squared_error: 3.0744 -
val_loss: 11.5970 - val_root_mean_squared_error: 3.4054
Epoch 90/200
72/72 - 0s - 794us/step - loss: 9.4256 - root_mean_squared_error: 3.0701 -
val_loss: 11.6501 - val_root_mean_squared_error: 3.4132
Epoch 91/200
72/72 - 0s - 813us/step - loss: 9.4069 - root_mean_squared_error: 3.0671 -
val_loss: 11.5931 - val_root_mean_squared_error: 3.4049
Epoch 92/200
72/72 - 0s - 796us/step - loss: 9.3899 - root_mean_squared_error: 3.0643 -
val_loss: 11.6229 - val_root_mean_squared_error: 3.4092
Epoch 93/200
72/72 - 0s - 804us/step - loss: 9.3780 - root_mean_squared_error: 3.0624 -
val_loss: 11.6525 - val_root_mean_squared_error: 3.4136
Epoch 94/200
72/72 - 0s - 809us/step - loss: 9.3597 - root_mean_squared_error: 3.0594 -
val_loss: 11.6707 - val_root_mean_squared_error: 3.4162
Epoch 95/200
72/72 - 0s - 799us/step - loss: 9.3402 - root_mean_squared_error: 3.0562 -
val_loss: 11.6622 - val_root_mean_squared_error: 3.4150
Epoch 96/200
72/72 - 0s - 809us/step - loss: 9.3118 - root_mean_squared_error: 3.0515 -
val_loss: 11.6912 - val_root_mean_squared_error: 3.4192
Epoch 97/200
72/72 - 0s - 803us/step - loss: 9.3014 - root_mean_squared_error: 3.0498 -
val_loss: 11.6844 - val_root_mean_squared_error: 3.4182
Epoch 98/200
72/72 - 0s - 803us/step - loss: 9.2856 - root_mean_squared_error: 3.0472 -
val_loss: 11.7221 - val_root_mean_squared_error: 3.4238
Epoch 99/200
72/72 - 0s - 798us/step - loss: 9.2663 - root_mean_squared_error: 3.0441 -
val_loss: 11.7260 - val_root_mean_squared_error: 3.4243
Epoch 100/200
72/72 - 0s - 800us/step - loss: 9.2492 - root_mean_squared_error: 3.0412 -
val_loss: 11.7452 - val_root_mean_squared_error: 3.4271
```

```
Epoch 101/200
72/72 - 0s - 792us/step - loss: 9.2164 - root_mean_squared_error: 3.0359 -
val_loss: 11.7364 - val_root_mean_squared_error: 3.4258
Epoch 102/200
72/72 - 0s - 803us/step - loss: 9.2151 - root_mean_squared_error: 3.0356 -
val_loss: 11.7840 - val_root_mean_squared_error: 3.4328
Epoch 103/200
72/72 - 0s - 803us/step - loss: 9.1860 - root_mean_squared_error: 3.0308 -
val_loss: 11.7662 - val_root_mean_squared_error: 3.4302
Epoch 104/200
72/72 - 0s - 799us/step - loss: 9.1760 - root_mean_squared_error: 3.0292 -
val_loss: 11.7878 - val_root_mean_squared_error: 3.4333
Epoch 105/200
72/72 - 0s - 795us/step - loss: 9.1549 - root_mean_squared_error: 3.0257 -
val_loss: 11.7798 - val_root_mean_squared_error: 3.4322
Epoch 106/200
72/72 - 0s - 791us/step - loss: 9.1399 - root_mean_squared_error: 3.0232 -
val_loss: 11.8466 - val_root_mean_squared_error: 3.4419
Epoch 107/200
72/72 - 0s - 797us/step - loss: 9.1230 - root_mean_squared_error: 3.0204 -
val_loss: 11.8313 - val_root_mean_squared_error: 3.4397
Epoch 108/200
72/72 - 0s - 810us/step - loss: 9.1087 - root_mean_squared_error: 3.0181 -
val_loss: 11.8220 - val_root_mean_squared_error: 3.4383
Epoch 109/200
72/72 - 0s - 794us/step - loss: 9.0882 - root_mean_squared_error: 3.0147 -
val_loss: 11.8415 - val_root_mean_squared_error: 3.4411
Epoch 110/200
72/72 - 0s - 811us/step - loss: 9.0687 - root_mean_squared_error: 3.0114 -
val_loss: 11.8636 - val_root_mean_squared_error: 3.4444
Epoch 111/200
72/72 - 0s - 802us/step - loss: 9.0601 - root_mean_squared_error: 3.0100 -
val_loss: 11.8636 - val_root_mean_squared_error: 3.4444
Epoch 112/200
72/72 - 0s - 799us/step - loss: 9.0377 - root_mean_squared_error: 3.0063 -
val_loss: 11.8955 - val_root_mean_squared_error: 3.4490
Epoch 113/200
72/72 - 0s - 807us/step - loss: 9.0157 - root_mean_squared_error: 3.0026 -
val_loss: 11.8966 - val_root_mean_squared_error: 3.4491
Epoch 114/200
72/72 - 0s - 799us/step - loss: 9.0016 - root_mean_squared_error: 3.0003 -
val_loss: 11.9325 - val_root_mean_squared_error: 3.4543
Epoch 115/200
72/72 - 0s - 785us/step - loss: 8.9893 - root_mean_squared_error: 2.9982 -
val_loss: 11.9220 - val_root_mean_squared_error: 3.4528
Epoch 116/200
72/72 - 0s - 800us/step - loss: 8.9825 - root_mean_squared_error: 2.9971 -
val_loss: 11.9541 - val_root_mean_squared_error: 3.4575
Epoch 117/200
72/72 - 0s - 797us/step - loss: 8.9566 - root_mean_squared_error: 2.9928 -
val_loss: 11.9175 - val_root_mean_squared_error: 3.4522
Epoch 118/200
72/72 - 0s - 845us/step - loss: 8.9489 - root_mean_squared_error: 2.9915 -
val_loss: 11.9629 - val_root_mean_squared_error: 3.4587
Epoch 119/200
72/72 - 0s - 797us/step - loss: 8.9317 - root_mean_squared_error: 2.9886 -
val_loss: 11.9568 - val_root_mean_squared_error: 3.4579
Epoch 120/200
72/72 - 0s - 807us/step - loss: 8.9039 - root_mean_squared_error: 2.9839 -
val_loss: 11.9676 - val_root_mean_squared_error: 3.4594
```

```
Epoch 121/200
72/72 - 0s - 792us/step - loss: 8.8958 - root_mean_squared_error: 2.9826 -
val_loss: 11.9826 - val_root_mean_squared_error: 3.4616
Epoch 122/200
72/72 - 0s - 786us/step - loss: 8.8759 - root_mean_squared_error: 2.9792 -
val_loss: 12.0326 - val_root_mean_squared_error: 3.4688
Epoch 123/200
72/72 - 0s - 797us/step - loss: 8.8545 - root_mean_squared_error: 2.9757 -
val_loss: 12.0600 - val_root_mean_squared_error: 3.4727
Epoch 124/200
72/72 - 0s - 808us/step - loss: 8.8518 - root_mean_squared_error: 2.9752 -
val_loss: 12.0569 - val_root_mean_squared_error: 3.4723
Epoch 125/200
72/72 - 0s - 857us/step - loss: 8.8450 - root_mean_squared_error: 2.9740 -
val_loss: 12.0752 - val_root_mean_squared_error: 3.4749
Epoch 126/200
72/72 - 0s - 797us/step - loss: 8.8209 - root_mean_squared_error: 2.9700 -
val_loss: 12.1111 - val_root_mean_squared_error: 3.4801
Epoch 127/200
72/72 - 0s - 803us/step - loss: 8.8070 - root_mean_squared_error: 2.9677 -
val_loss: 12.0582 - val_root_mean_squared_error: 3.4725
Epoch 128/200
72/72 - 0s - 802us/step - loss: 8.7821 - root_mean_squared_error: 2.9635 -
val_loss: 12.1222 - val_root_mean_squared_error: 3.4817
Epoch 129/200
72/72 - 0s - 807us/step - loss: 8.7802 - root_mean_squared_error: 2.9631 -
val_loss: 12.1288 - val_root_mean_squared_error: 3.4826
Epoch 130/200
72/72 - 0s - 797us/step - loss: 8.7707 - root_mean_squared_error: 2.9615 -
val_loss: 12.1266 - val_root_mean_squared_error: 3.4823
Epoch 131/200
72/72 - 0s - 798us/step - loss: 8.7508 - root_mean_squared_error: 2.9582 -
val_loss: 12.1290 - val_root_mean_squared_error: 3.4827
Epoch 132/200
72/72 - 0s - 829us/step - loss: 8.7285 - root_mean_squared_error: 2.9544 -
val_loss: 12.1647 - val_root_mean_squared_error: 3.4878
Epoch 133/200
72/72 - 0s - 1ms/step - loss: 8.7084 - root_mean_squared_error: 2.9510 - v
al_loss: 12.2170 - val_root_mean_squared_error: 3.4953
Epoch 134/200
72/72 - 0s - 785us/step - loss: 8.6996 - root_mean_squared_error: 2.9495 -
val_loss: 12.1722 - val_root_mean_squared_error: 3.4889
Epoch 135/200
72/72 - 0s - 796us/step - loss: 8.6818 - root_mean_squared_error: 2.9465 -
val_loss: 12.1982 - val_root_mean_squared_error: 3.4926
Epoch 136/200
72/72 - 0s - 799us/step - loss: 8.6721 - root_mean_squared_error: 2.9448 -
val_loss: 12.2101 - val_root_mean_squared_error: 3.4943
Epoch 137/200
72/72 - 0s - 801us/step - loss: 8.6454 - root_mean_squared_error: 2.9403 -
val_loss: 12.1854 - val_root_mean_squared_error: 3.4908
Epoch 138/200
72/72 - 0s - 814us/step - loss: 8.6293 - root_mean_squared_error: 2.9376 -
val_loss: 12.2486 - val_root_mean_squared_error: 3.4998
Epoch 139/200
72/72 - 0s - 795us/step - loss: 8.6136 - root_mean_squared_error: 2.9349 -
val_loss: 12.2304 - val_root_mean_squared_error: 3.4972
Epoch 140/200
72/72 - 0s - 792us/step - loss: 8.5942 - root_mean_squared_error: 2.9316 -
val_loss: 12.2794 - val_root_mean_squared_error: 3.5042
```

```
Epoch 141/200
72/72 - 0s - 793us/step - loss: 8.5856 - root_mean_squared_error: 2.9301 -
val_loss: 12.2523 - val_root_mean_squared_error: 3.5003
Epoch 142/200
72/72 - 0s - 806us/step - loss: 8.5673 - root_mean_squared_error: 2.9270 -
val_loss: 12.2688 - val_root_mean_squared_error: 3.5027
Epoch 143/200
72/72 - 0s - 802us/step - loss: 8.5616 - root_mean_squared_error: 2.9260 -
val_loss: 12.2973 - val_root_mean_squared_error: 3.5067
Epoch 144/200
72/72 - 0s - 806us/step - loss: 8.5279 - root_mean_squared_error: 2.9203 -
val_loss: 12.2947 - val_root_mean_squared_error: 3.5064
Epoch 145/200
72/72 - 0s - 822us/step - loss: 8.5227 - root_mean_squared_error: 2.9194 -
val_loss: 12.3298 - val_root_mean_squared_error: 3.5114
Epoch 146/200
72/72 - 0s - 801us/step - loss: 8.5037 - root_mean_squared_error: 2.9161 -
val_loss: 12.3466 - val_root_mean_squared_error: 3.5138
Epoch 147/200
72/72 - 0s - 797us/step - loss: 8.4993 - root_mean_squared_error: 2.9153 -
val_loss: 12.3597 - val_root_mean_squared_error: 3.5156
Epoch 148/200
72/72 - 0s - 802us/step - loss: 8.4753 - root_mean_squared_error: 2.9112 -
val_loss: 12.3839 - val_root_mean_squared_error: 3.5191
Epoch 149/200
72/72 - 0s - 791us/step - loss: 8.4735 - root_mean_squared_error: 2.9109 -
val_loss: 12.3939 - val_root_mean_squared_error: 3.5205
Epoch 150/200
72/72 - 0s - 798us/step - loss: 8.4410 - root_mean_squared_error: 2.9053 -
val_loss: 12.4055 - val_root_mean_squared_error: 3.5221
Epoch 151/200
72/72 - 0s - 811us/step - loss: 8.4315 - root_mean_squared_error: 2.9037 -
val_loss: 12.4203 - val_root_mean_squared_error: 3.5242
Epoch 152/200
72/72 - 0s - 808us/step - loss: 8.4090 - root_mean_squared_error: 2.8998 -
val_loss: 12.4373 - val_root_mean_squared_error: 3.5267
Epoch 153/200
72/72 - 0s - 802us/step - loss: 8.4019 - root_mean_squared_error: 2.8986 -
val_loss: 12.4453 - val_root_mean_squared_error: 3.5278
Epoch 154/200
72/72 - 0s - 800us/step - loss: 8.3866 - root_mean_squared_error: 2.8960 -
val_loss: 12.4445 - val_root_mean_squared_error: 3.5277
Epoch 155/200
72/72 - 0s - 804us/step - loss: 8.3633 - root_mean_squared_error: 2.8919 -
val_loss: 12.5134 - val_root_mean_squared_error: 3.5374
Epoch 156/200
72/72 - 0s - 811us/step - loss: 8.3632 - root_mean_squared_error: 2.8919 -
val_loss: 12.5002 - val_root_mean_squared_error: 3.5356
Epoch 157/200
72/72 - 0s - 805us/step - loss: 8.3305 - root_mean_squared_error: 2.8863 -
val_loss: 12.5097 - val_root_mean_squared_error: 3.5369
Epoch 158/200
72/72 - 0s - 799us/step - loss: 8.3192 - root_mean_squared_error: 2.8843 -
val_loss: 12.5276 - val_root_mean_squared_error: 3.5394
Epoch 159/200
72/72 - 0s - 807us/step - loss: 8.3060 - root_mean_squared_error: 2.8820 -
val_loss: 12.5318 - val_root_mean_squared_error: 3.5400
Epoch 160/200
72/72 - 0s - 796us/step - loss: 8.2772 - root_mean_squared_error: 2.8770 -
val_loss: 12.5557 - val_root_mean_squared_error: 3.5434
```

```
Epoch 161/200
72/72 - 0s - 796us/step - loss: 8.2734 - root_mean_squared_error: 2.8764 -
val_loss: 12.5753 - val_root_mean_squared_error: 3.5462
Epoch 162/200
72/72 - 0s - 821us/step - loss: 8.2520 - root_mean_squared_error: 2.8726 -
val_loss: 12.5977 - val_root_mean_squared_error: 3.5493
Epoch 163/200
72/72 - 0s - 800us/step - loss: 8.2365 - root_mean_squared_error: 2.8699 -
val_loss: 12.5715 - val_root_mean_squared_error: 3.5456
Epoch 164/200
72/72 - 0s - 797us/step - loss: 8.2180 - root_mean_squared_error: 2.8667 -
val_loss: 12.6244 - val_root_mean_squared_error: 3.5531
Epoch 165/200
72/72 - 0s - 804us/step - loss: 8.1998 - root_mean_squared_error: 2.8635 -
val_loss: 12.6137 - val_root_mean_squared_error: 3.5516
Epoch 166/200
72/72 - 0s - 807us/step - loss: 8.1994 - root_mean_squared_error: 2.8635 -
val_loss: 12.6199 - val_root_mean_squared_error: 3.5525
Epoch 167/200
72/72 - 0s - 802us/step - loss: 8.1776 - root_mean_squared_error: 2.8596 -
val_loss: 12.6406 - val_root_mean_squared_error: 3.5554
Epoch 168/200
72/72 - 0s - 810us/step - loss: 8.1568 - root_mean_squared_error: 2.8560 -
val_loss: 12.6503 - val_root_mean_squared_error: 3.5567
Epoch 169/200
72/72 - 0s - 790us/step - loss: 8.1437 - root_mean_squared_error: 2.8537 -
val_loss: 12.6588 - val_root_mean_squared_error: 3.5579
Epoch 170/200
72/72 - 0s - 800us/step - loss: 8.1312 - root_mean_squared_error: 2.8515 -
val_loss: 12.6906 - val_root_mean_squared_error: 3.5624
Epoch 171/200
72/72 - 0s - 800us/step - loss: 8.1182 - root_mean_squared_error: 2.8492 -
val_loss: 12.7247 - val_root_mean_squared_error: 3.5672
Epoch 172/200
72/72 - 0s - 811us/step - loss: 8.1022 - root_mean_squared_error: 2.8464 -
val_loss: 12.7105 - val_root_mean_squared_error: 3.5652
Epoch 173/200
72/72 - 0s - 800us/step - loss: 8.0835 - root_mean_squared_error: 2.8431 -
val_loss: 12.7025 - val_root_mean_squared_error: 3.5641
Epoch 174/200
72/72 - 0s - 808us/step - loss: 8.0649 - root_mean_squared_error: 2.8399 -
val_loss: 12.7378 - val_root_mean_squared_error: 3.5690
Epoch 175/200
72/72 - 0s - 801us/step - loss: 8.0446 - root_mean_squared_error: 2.8363 -
val_loss: 12.7509 - val_root_mean_squared_error: 3.5708
Epoch 176/200
72/72 - 0s - 806us/step - loss: 8.0340 - root_mean_squared_error: 2.8344 -
val_loss: 12.7543 - val_root_mean_squared_error: 3.5713
Epoch 177/200
72/72 - 0s - 814us/step - loss: 8.0249 - root_mean_squared_error: 2.8328 -
val_loss: 12.7971 - val_root_mean_squared_error: 3.5773
Epoch 178/200
72/72 - 0s - 810us/step - loss: 8.0101 - root_mean_squared_error: 2.8302 -
val_loss: 12.8158 - val_root_mean_squared_error: 3.5799
Epoch 179/200
72/72 - 0s - 807us/step - loss: 7.9927 - root_mean_squared_error: 2.8271 -
val_loss: 12.8289 - val_root_mean_squared_error: 3.5818
Epoch 180/200
72/72 - 0s - 805us/step - loss: 7.9717 - root_mean_squared_error: 2.8234 -
val_loss: 12.8321 - val_root_mean_squared_error: 3.5822
```

```
Epoch 181/200
72/72 - 0s - 825us/step - loss: 7.9531 - root_mean_squared_error: 2.8201 -
val_loss: 12.8278 - val_root_mean_squared_error: 3.5816
Epoch 182/200
72/72 - 0s - 805us/step - loss: 7.9414 - root_mean_squared_error: 2.8181 -
val_loss: 12.8452 - val_root_mean_squared_error: 3.5840
Epoch 183/200
72/72 - 0s - 803us/step - loss: 7.9227 - root_mean_squared_error: 2.8147 -
val_loss: 12.8956 - val_root_mean_squared_error: 3.5911
Epoch 184/200
72/72 - 0s - 806us/step - loss: 7.9128 - root_mean_squared_error: 2.8130 -
val_loss: 12.9347 - val_root_mean_squared_error: 3.5965
Epoch 185/200
72/72 - 0s - 804us/step - loss: 7.9011 - root_mean_squared_error: 2.8109 -
val_loss: 12.9107 - val_root_mean_squared_error: 3.5931
Epoch 186/200
72/72 - 0s - 812us/step - loss: 7.8714 - root_mean_squared_error: 2.8056 -
val_loss: 12.9152 - val_root_mean_squared_error: 3.5938
Epoch 187/200
72/72 - 0s - 792us/step - loss: 7.8687 - root_mean_squared_error: 2.8051 -
val_loss: 12.9114 - val_root_mean_squared_error: 3.5932
Epoch 188/200
72/72 - 0s - 806us/step - loss: 7.8443 - root_mean_squared_error: 2.8008 -
val_loss: 12.9450 - val_root_mean_squared_error: 3.5979
Epoch 189/200
72/72 - 0s - 793us/step - loss: 7.8307 - root_mean_squared_error: 2.7983 -
val_loss: 12.9502 - val_root_mean_squared_error: 3.5986
Epoch 190/200
72/72 - 0s - 805us/step - loss: 7.8167 - root_mean_squared_error: 2.7958 -
val_loss: 13.0105 - val_root_mean_squared_error: 3.6070
Epoch 191/200
72/72 - 0s - 806us/step - loss: 7.8031 - root_mean_squared_error: 2.7934 -
val_loss: 13.0305 - val_root_mean_squared_error: 3.6098
Epoch 192/200
72/72 - 0s - 812us/step - loss: 7.7871 - root_mean_squared_error: 2.7905 -
val_loss: 13.0171 - val_root_mean_squared_error: 3.6079
Epoch 193/200
72/72 - 0s - 798us/step - loss: 7.7577 - root_mean_squared_error: 2.7853 -
val_loss: 13.0707 - val_root_mean_squared_error: 3.6153
Epoch 194/200
72/72 - 0s - 803us/step - loss: 7.7610 - root_mean_squared_error: 2.7859 -
val_loss: 13.0510 - val_root_mean_squared_error: 3.6126
Epoch 195/200
72/72 - 0s - 814us/step - loss: 7.7315 - root_mean_squared_error: 2.7806 -
val_loss: 13.0744 - val_root_mean_squared_error: 3.6159
Epoch 196/200
72/72 - 0s - 809us/step - loss: 7.7164 - root_mean_squared_error: 2.7778 -
val_loss: 13.1017 - val_root_mean_squared_error: 3.6196
Epoch 197/200
72/72 - 0s - 819us/step - loss: 7.7029 - root_mean_squared_error: 2.7754 -
val_loss: 13.1263 - val_root_mean_squared_error: 3.6230
Epoch 198/200
72/72 - 0s - 804us/step - loss: 7.6885 - root_mean_squared_error: 2.7728 -
val_loss: 13.1046 - val_root_mean_squared_error: 3.6200
Epoch 199/200
72/72 - 0s - 820us/step - loss: 7.6720 - root_mean_squared_error: 2.7698 -
val_loss: 13.1369 - val_root_mean_squared_error: 3.6245
Epoch 200/200
72/72 - 0s - 809us/step - loss: 7.6535 - root_mean_squared_error: 2.7665 -
val_loss: 13.1322 - val_root_mean_squared_error: 3.6238
```

In [70]:
```
%load_ext tensorboard
%tensorboard --logdir logs/fit
```

In [71]:
```
model.save('my_tuned_weather_model.keras')
```

Advanced stuff

In [72]:
```
import pathlib
import numpy as np

DATA_FOLDER = pathlib.Path('data_dollar/') # change to location where you
```

```
train_images = np.load(DATA_FOLDER / 'train_images.npy')
val_images = np.load(DATA_FOLDER / 'test_images.npy')
train_labels = np.load(DATA_FOLDER / 'train_labels.npy')
val_labels = np.load(DATA_FOLDER / 'test_labels.npy')
```

In [73]:
```
train_images.shape
```

Out[73]: (878, 64, 64, 3)

In [74]:
```
train_images.min(), train_images.max()
```

Out[74]: (0, 255)

In [75]:
```
train_labels.shape
```

Out[75]: (878,)

In [76]:
```
train_labels.min(), train_labels.max()
```

Out[76]: (0, 9)

In [77]:
```
train_images = train_images / 255.0
val_images = val_images / 255.0
```

In [78]:
```
inputs = keras.Input(shape=train_images.shape[1:])
x = keras.layers.Conv2D(50, (3, 3), activation='relu')(inputs)
x = keras.layers.Conv2D(50, (3, 3), activation='relu')(x)
x = keras.layers.Flatten()(x)
outputs = keras.layers.Dense(10)(x)

model = keras.Model(inputs=inputs, outputs=outputs, name="dollar_street_m

model.summary()
```

Model: "dollar_street_model_small"

Layer (type)	Output Shape	P
input_layer_1 (InputLayer)	(None, 64, 64, 3)	
conv2d (Conv2D)	(None, 62, 62, 50)	
conv2d_1 (Conv2D)	(None, 60, 60, 50)	
flatten (Flatten)	(None, 180000)	
dense_17 (Dense)	(None, 10)	1,8

Total params: 1,823,960 (6.96 MB)
Trainable params: 1,823,960 (6.96 MB)
Non-trainable params: 0 (0.00 B)

In [79]:
```
def create_nn():
    inputs = keras.Input(shape=train_images.shape[1:])
    x = keras.layers.Conv2D(50, (3, 3), activation='relu')(inputs)
    x = keras.layers.MaxPooling2D((2, 2))(x) # a new maxpooling layer
    x = keras.layers.Conv2D(50, (3, 3), activation='relu')(x)
    x = keras.layers.MaxPooling2D((2, 2))(x) # a new maxpooling layer (sa
```

```
    x = keras.layers.Flatten()(x)
    x = keras.layers.Dense(50, activation='relu')(x) # a new Dense layer
    outputs = keras.layers.Dense(10)(x)

    model = keras.Model(inputs=inputs, outputs=outputs, name="dollar_stre
    return model

model = create_nn()
model.summary()
```

Model: "dollar_street_model"

Layer (type)	Output Shape	P
input_layer_2 (InputLayer)	(None, 64, 64, 3)	
conv2d_2 (Conv2D)	(None, 62, 62, 50)	
max_pooling2d (MaxPooling2D)	(None, 31, 31, 50)	
conv2d_3 (Conv2D)	(None, 29, 29, 50)	
max_pooling2d_1 (MaxPooling2D)	(None, 14, 14, 50)	
flatten_1 (Flatten)	(None, 9800)	
dense_18 (Dense)	(None, 50)	4
dense_19 (Dense)	(None, 10)	

Total params: 514,510 (1.96 MB)

Trainable params: 514,510 (1.96 MB)

Non-trainable params: 0 (0.00 B)

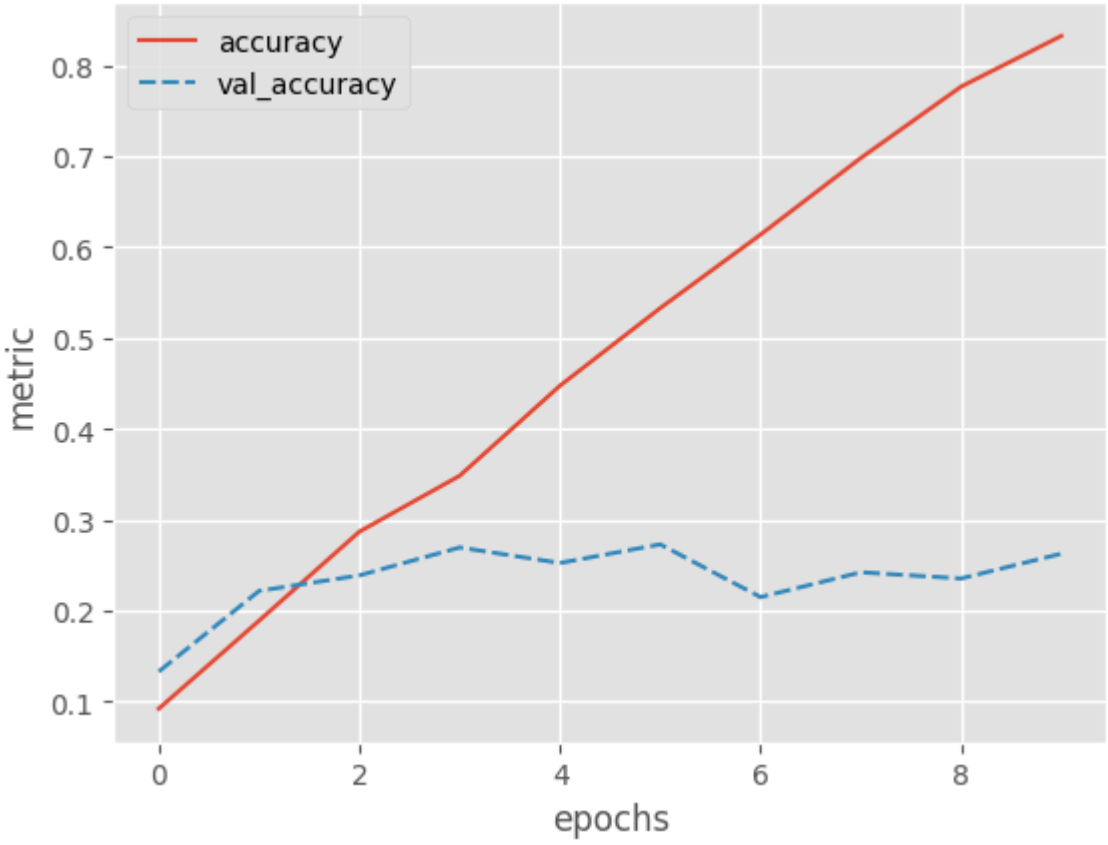
In [80]:
```
def compile_model(model):
    model.compile(optimizer='adam',
                  loss=keras.losses.SparseCategoricalCrossentropy(from_lo
                  metrics=['accuracy'])
compile_model(model)
```

In [81]:
```
history = model.fit(train_images, train_labels, epochs=10,
                    validation_data=(val_images, val_labels))
```

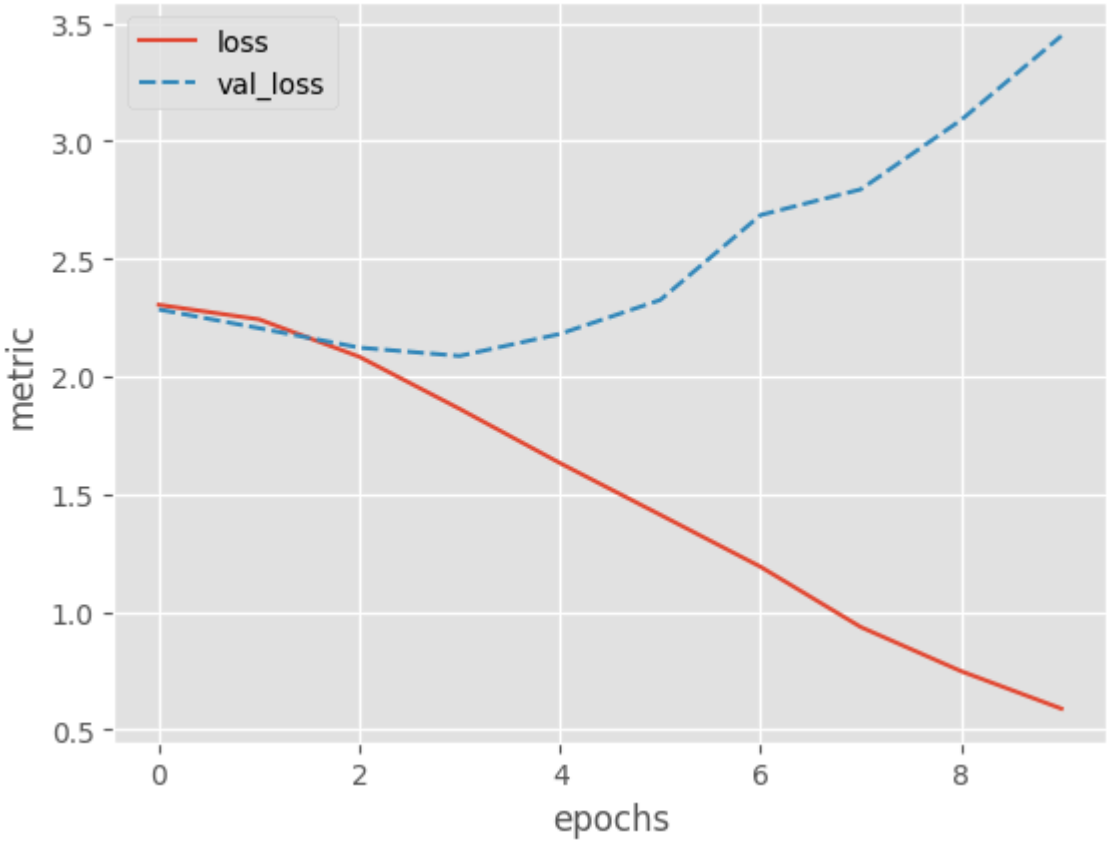
```
Epoch 1/10
28/28 ━━━━━━━━━━━━━━━━━━━━ 1s 40ms/step - accuracy: 0.0865 - loss: 2.3160
- val_accuracy: 0.1331 - val_loss: 2.2852
Epoch 2/10
28/28 ━━━━━━━━━━━━━━━━━━━━ 1s 37ms/step - accuracy: 0.1852 - loss: 2.2555
- val_accuracy: 0.2218 - val_loss: 2.2056
Epoch 3/10
28/28 ━━━━━━━━━━━━━━━━━━━━ 1s 38ms/step - accuracy: 0.3079 - loss: 2.0997
- val_accuracy: 0.2389 - val_loss: 2.1230
Epoch 4/10
28/28 ━━━━━━━━━━━━━━━━━━━━ 1s 39ms/step - accuracy: 0.3645 - loss: 1.8547
- val_accuracy: 0.2696 - val_loss: 2.0877
Epoch 5/10
28/28 ━━━━━━━━━━━━━━━━━━━━ 1s 38ms/step - accuracy: 0.4712 - loss: 1.6175
- val_accuracy: 0.2526 - val_loss: 2.1813
Epoch 6/10
28/28 ━━━━━━━━━━━━━━━━━━━━ 1s 38ms/step - accuracy: 0.5471 - loss: 1.4167
- val_accuracy: 0.2730 - val_loss: 2.3254
Epoch 7/10
28/28 ━━━━━━━━━━━━━━━━━━━━ 1s 38ms/step - accuracy: 0.6051 - loss: 1.1906
- val_accuracy: 0.2150 - val_loss: 2.6861
Epoch 8/10
28/28 ━━━━━━━━━━━━━━━━━━━━ 1s 38ms/step - accuracy: 0.7083 - loss: 0.9115
- val_accuracy: 0.2423 - val_loss: 2.7954
Epoch 9/10
28/28 ━━━━━━━━━━━━━━━━━━━━ 1s 38ms/step - accuracy: 0.7796 - loss: 0.7237
- val_accuracy: 0.2355 - val_loss: 3.0900
Epoch 10/10
28/28 ━━━━━━━━━━━━━━━━━━━━ 1s 38ms/step - accuracy: 0.8383 - loss: 0.5766
- val_accuracy: 0.2628 - val_loss: 3.4472
```

In [82]:

```python
import seaborn as sns
import matplotlib.pyplot as plt
import pandas as pd

def plot_history(history, metrics):
    """
    Plot the training history

    Args:
        history (keras History object that is returned by model.fit())
        metrics(str, list): Metric or a list of metrics to plot
    """
    history_df = pd.DataFrame.from_dict(history.history)
    sns.lineplot(data=history_df[metrics])
    plt.xlabel("epochs")
    plt.ylabel("metric")
plot_history(history, ['accuracy', 'val_accuracy'])
```



In [83]:
```
plot_history(history, ['loss', 'val_loss'])
```



In [84]:
```
def create_dense_model():
    inputs = keras.Input(shape=train_images.shape[1:])
    x = keras.layers.Flatten()(inputs)
    x = keras.layers.Dense(50, activation='relu')(x)
    x = keras.layers.Dense(50, activation='relu')(x)
```

```
    outputs = keras.layers.Dense(10)(x)
    return keras.models.Model(inputs=inputs, outputs=outputs,
                              name='dense_model')

dense_model = create_dense_model()
dense_model.summary()
```

Model: "dense_model"

Layer (type)	Output Shape	P
input_layer_3 (InputLayer)	(None, 64, 64, 3)	
flatten_2 (Flatten)	(None, 12288)	
dense_20 (Dense)	(None, 50)	6
dense_21 (Dense)	(None, 50)	
dense_22 (Dense)	(None, 10)	

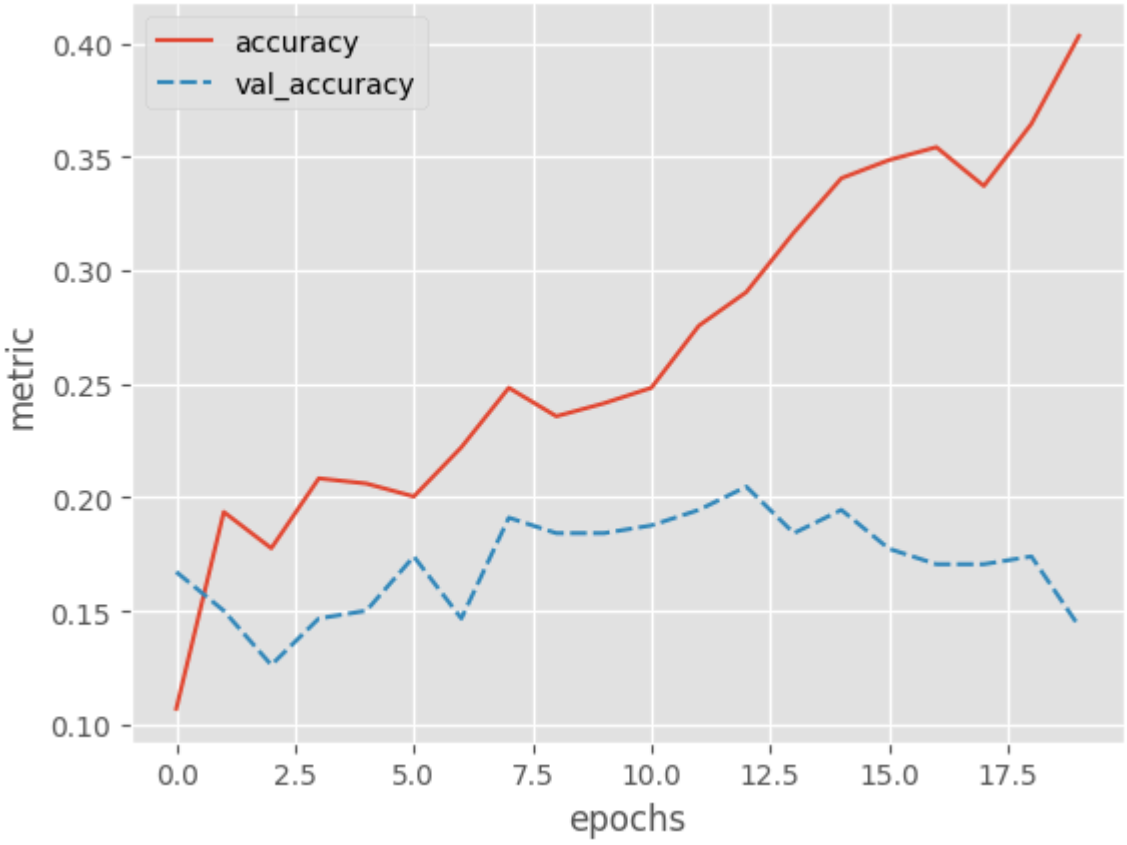
Total params: 617,510 (2.36 MB)

Trainable params: 617,510 (2.36 MB)

Non-trainable params: 0 (0.00 B)

In [85]:
```
compile_model(dense_model)
history = dense_model.fit(train_images, train_labels, epochs=20,
                          validation_data=(val_images, val_labels))
plot_history(history, ['accuracy', 'val_accuracy'])
```

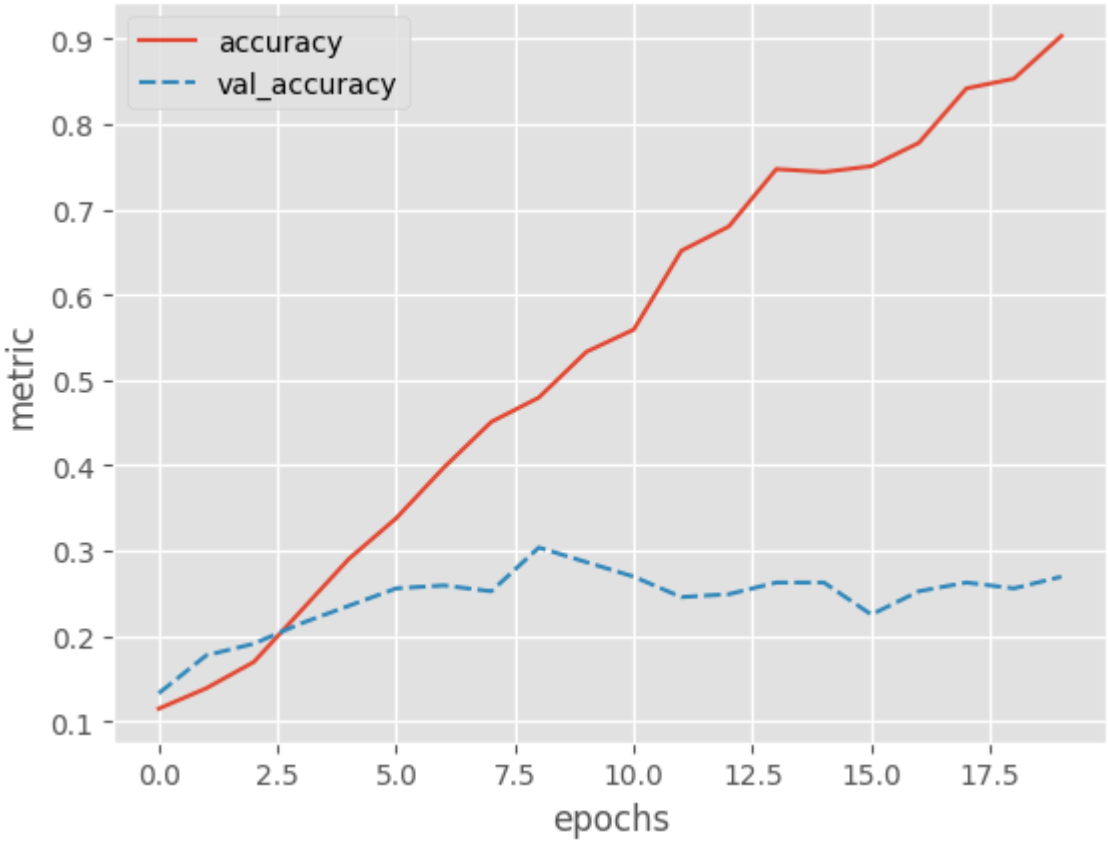
```
Epoch 1/20
28/28 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.1023 - loss: 3.0312 -
val_accuracy: 0.1672 - val_loss: 2.4178
Epoch 2/20
28/28 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.1941 - loss: 2.2829 -
val_accuracy: 0.1502 - val_loss: 2.2633
Epoch 3/20
28/28 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.1698 - loss: 2.2368 -
val_accuracy: 0.1263 - val_loss: 2.3077
Epoch 4/20
28/28 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.2084 - loss: 2.2093 -
val_accuracy: 0.1468 - val_loss: 2.3671
Epoch 5/20
28/28 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.2027 - loss: 2.1870 -
val_accuracy: 0.1502 - val_loss: 2.3300
Epoch 6/20
28/28 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.2189 - loss: 2.1832 -
val_accuracy: 0.1741 - val_loss: 2.2633
Epoch 7/20
28/28 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.2283 - loss: 2.1414 -
val_accuracy: 0.1468 - val_loss: 2.3366
Epoch 8/20
28/28 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.2539 - loss: 2.1473 -
val_accuracy: 0.1911 - val_loss: 2.2950
Epoch 9/20
28/28 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.2439 - loss: 2.0990 -
val_accuracy: 0.1843 - val_loss: 2.3382
Epoch 10/20
28/28 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.2386 - loss: 2.1363 -
val_accuracy: 0.1843 - val_loss: 2.2636
Epoch 11/20
28/28 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.2599 - loss: 2.0731 -
val_accuracy: 0.1877 - val_loss: 2.2314
Epoch 12/20
28/28 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.2755 - loss: 2.0248 -
val_accuracy: 0.1945 - val_loss: 2.2562
Epoch 13/20
28/28 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.3142 - loss: 1.9526 -
val_accuracy: 0.2048 - val_loss: 2.2616
Epoch 14/20
28/28 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.3200 - loss: 1.9231 -
val_accuracy: 0.1843 - val_loss: 2.2448
Epoch 15/20
28/28 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.3562 - loss: 1.8629 -
val_accuracy: 0.1945 - val_loss: 2.2878
Epoch 16/20
28/28 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.3525 - loss: 1.8459 -
val_accuracy: 0.1775 - val_loss: 2.3664
Epoch 17/20
28/28 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.3759 - loss: 1.8201 -
val_accuracy: 0.1706 - val_loss: 2.4016
Epoch 18/20
28/28 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.3559 - loss: 1.8562 -
val_accuracy: 0.1706 - val_loss: 2.3895
Epoch 19/20
28/28 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.3812 - loss: 1.7838 -
val_accuracy: 0.1741 - val_loss: 2.4133
Epoch 20/20
28/28 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.4120 - loss: 1.7165 -
val_accuracy: 0.1433 - val_loss: 2.4999
```



In [86]:
```
def create_nn_extra_layer():
    inputs = keras.Input(shape=train_images.shape[1:])
    x = keras.layers.Conv2D(50, (3, 3), activation='relu')(inputs)
    x = keras.layers.MaxPooling2D((2, 2))(x)
    x = keras.layers.Conv2D(50, (3, 3), activation='relu')(x)
    x = keras.layers.MaxPooling2D((2, 2))(x) #
    x = keras.layers.Conv2D(50, (3, 3), activation='relu')(x) # extra lay
    x = keras.layers.MaxPooling2D((2, 2))(x) # extra layer
    x = keras.layers.Flatten()(x)
    x = keras.layers.Dense(50, activation='relu')(x)
    outputs = keras.layers.Dense(10)(x)

    model = keras.Model(inputs=inputs, outputs=outputs, name="dollar_stre
    return model

model = create_nn_extra_layer()
```

In [87]:
```
compile_model(model)
history = model.fit(train_images, train_labels, epochs=20,
                    validation_data=(val_images, val_labels))
plot_history(history, ['accuracy', 'val_accuracy'])
```

```
Epoch 1/20
28/28 ━━━━━━━━━━━━━━━━━━━━ 2s 45ms/step - accuracy: 0.1133 - loss: 2.3125
- val_accuracy: 0.1331 - val_loss: 2.2897
Epoch 2/20
28/28 ━━━━━━━━━━━━━━━━━━━━ 1s 41ms/step - accuracy: 0.1354 - loss: 2.2967
- val_accuracy: 0.1775 - val_loss: 2.2694
Epoch 3/20
28/28 ━━━━━━━━━━━━━━━━━━━━ 1s 41ms/step - accuracy: 0.1747 - loss: 2.2663
- val_accuracy: 0.1911 - val_loss: 2.2270
Epoch 4/20
28/28 ━━━━━━━━━━━━━━━━━━━━ 1s 41ms/step - accuracy: 0.2430 - loss: 2.1748
- val_accuracy: 0.2150 - val_loss: 2.1737
Epoch 5/20
28/28 ━━━━━━━━━━━━━━━━━━━━ 1s 41ms/step - accuracy: 0.2926 - loss: 2.0296
- val_accuracy: 0.2355 - val_loss: 2.2166
Epoch 6/20
28/28 ━━━━━━━━━━━━━━━━━━━━ 1s 42ms/step - accuracy: 0.3477 - loss: 1.8808
- val_accuracy: 0.2560 - val_loss: 2.1921
Epoch 7/20
28/28 ━━━━━━━━━━━━━━━━━━━━ 1s 41ms/step - accuracy: 0.3942 - loss: 1.7550
- val_accuracy: 0.2594 - val_loss: 2.2898
Epoch 8/20
28/28 ━━━━━━━━━━━━━━━━━━━━ 1s 41ms/step - accuracy: 0.4392 - loss: 1.6020
- val_accuracy: 0.2526 - val_loss: 2.3869
Epoch 9/20
28/28 ━━━━━━━━━━━━━━━━━━━━ 1s 41ms/step - accuracy: 0.4687 - loss: 1.4814
- val_accuracy: 0.3038 - val_loss: 2.3680
Epoch 10/20
28/28 ━━━━━━━━━━━━━━━━━━━━ 1s 41ms/step - accuracy: 0.5419 - loss: 1.3312
- val_accuracy: 0.2867 - val_loss: 2.4628
Epoch 11/20
28/28 ━━━━━━━━━━━━━━━━━━━━ 1s 42ms/step - accuracy: 0.5649 - loss: 1.2087
- val_accuracy: 0.2696 - val_loss: 2.6253
Epoch 12/20
28/28 ━━━━━━━━━━━━━━━━━━━━ 1s 41ms/step - accuracy: 0.6485 - loss: 1.0625
- val_accuracy: 0.2457 - val_loss: 2.8272
Epoch 13/20
28/28 ━━━━━━━━━━━━━━━━━━━━ 1s 40ms/step - accuracy: 0.6677 - loss: 0.9343
- val_accuracy: 0.2491 - val_loss: 2.9606
Epoch 14/20
28/28 ━━━━━━━━━━━━━━━━━━━━ 1s 41ms/step - accuracy: 0.7516 - loss: 0.7908
- val_accuracy: 0.2628 - val_loss: 3.1461
Epoch 15/20
28/28 ━━━━━━━━━━━━━━━━━━━━ 1s 41ms/step - accuracy: 0.7552 - loss: 0.6696
- val_accuracy: 0.2628 - val_loss: 3.3305
Epoch 16/20
28/28 ━━━━━━━━━━━━━━━━━━━━ 1s 42ms/step - accuracy: 0.7527 - loss: 0.6892
- val_accuracy: 0.2253 - val_loss: 3.6154
Epoch 17/20
28/28 ━━━━━━━━━━━━━━━━━━━━ 1s 41ms/step - accuracy: 0.7674 - loss: 0.6902
- val_accuracy: 0.2526 - val_loss: 3.7069
Epoch 18/20
28/28 ━━━━━━━━━━━━━━━━━━━━ 1s 41ms/step - accuracy: 0.8199 - loss: 0.5441
- val_accuracy: 0.2628 - val_loss: 3.9436
Epoch 19/20
28/28 ━━━━━━━━━━━━━━━━━━━━ 1s 41ms/step - accuracy: 0.8417 - loss: 0.4395
- val_accuracy: 0.2560 - val_loss: 4.0349
Epoch 20/20
28/28 ━━━━━━━━━━━━━━━━━━━━ 1s 41ms/step - accuracy: 0.9131 - loss: 0.3273
- val_accuracy: 0.2696 - val_loss: 4.2194
```



In [88]:
```
model.summary()
```

Model: "dollar_street_model"

Layer (type)	Output Shape	P
input_layer_4 (InputLayer)	(None, 64, 64, 3)	
conv2d_4 (Conv2D)	(None, 62, 62, 50)	
max_pooling2d_2 (MaxPooling2D)	(None, 31, 31, 50)	
conv2d_5 (Conv2D)	(None, 29, 29, 50)	
max_pooling2d_3 (MaxPooling2D)	(None, 14, 14, 50)	
conv2d_6 (Conv2D)	(None, 12, 12, 50)	
max_pooling2d_4 (MaxPooling2D)	(None, 6, 6, 50)	
flatten_3 (Flatten)	(None, 1800)	
dense_23 (Dense)	(None, 50)	
dense_24 (Dense)	(None, 10)	

Total params: 411,182 (1.57 MB)
Trainable params: 137,060 (535.39 KB)
Non-trainable params: 0 (0.00 B)
Optimizer params: 274,122 (1.05 MB)

In [92]:
```
def create_nn_with_dropout():
    inputs = keras.Input(shape=train_images.shape[1:])
```

```
    x = keras.layers.Conv2D(50, (3, 3), activation='relu')(inputs)
    x = keras.layers.MaxPooling2D((2, 2))(x)

    x = keras.layers.Conv2D(50, (3, 3), activation='relu')(x)
    x = keras.layers.MaxPooling2D((2, 2))(x)

    x = keras.layers.Conv2D(50, (3, 3), activation='relu')(x)
    x = keras.layers.MaxPooling2D((2, 2))(x)
    x = keras.layers.Dropout(0.5)(x) # This is new!

    x = keras.layers.Flatten()(x)
    x = keras.layers.Dense(50, activation='relu')(x)
    outputs = keras.layers.Dense(10)(x)
    model = keras.Model(inputs=inputs, outputs=outputs, name="dropout_mod
    return model

model_dropout = create_nn_with_dropout()
model_dropout.summary()
```

Model: "dropout_model"

Layer (type)	Output Shape	P
input_layer_6 (InputLayer)	(None, 64, 64, 3)	
conv2d_10 (Conv2D)	(None, 62, 62, 50)	
max_pooling2d_8 (MaxPooling2D)	(None, 31, 31, 50)	
conv2d_11 (Conv2D)	(None, 29, 29, 50)	
max_pooling2d_9 (MaxPooling2D)	(None, 14, 14, 50)	
conv2d_12 (Conv2D)	(None, 12, 12, 50)	
max_pooling2d_10 (MaxPooling2D)	(None, 6, 6, 50)	
dropout_1 (Dropout)	(None, 6, 6, 50)	
flatten_5 (Flatten)	(None, 1800)	
dense_27 (Dense)	(None, 50)	
dense_28 (Dense)	(None, 10)	

Total params: 137,060 (535.39 KB)

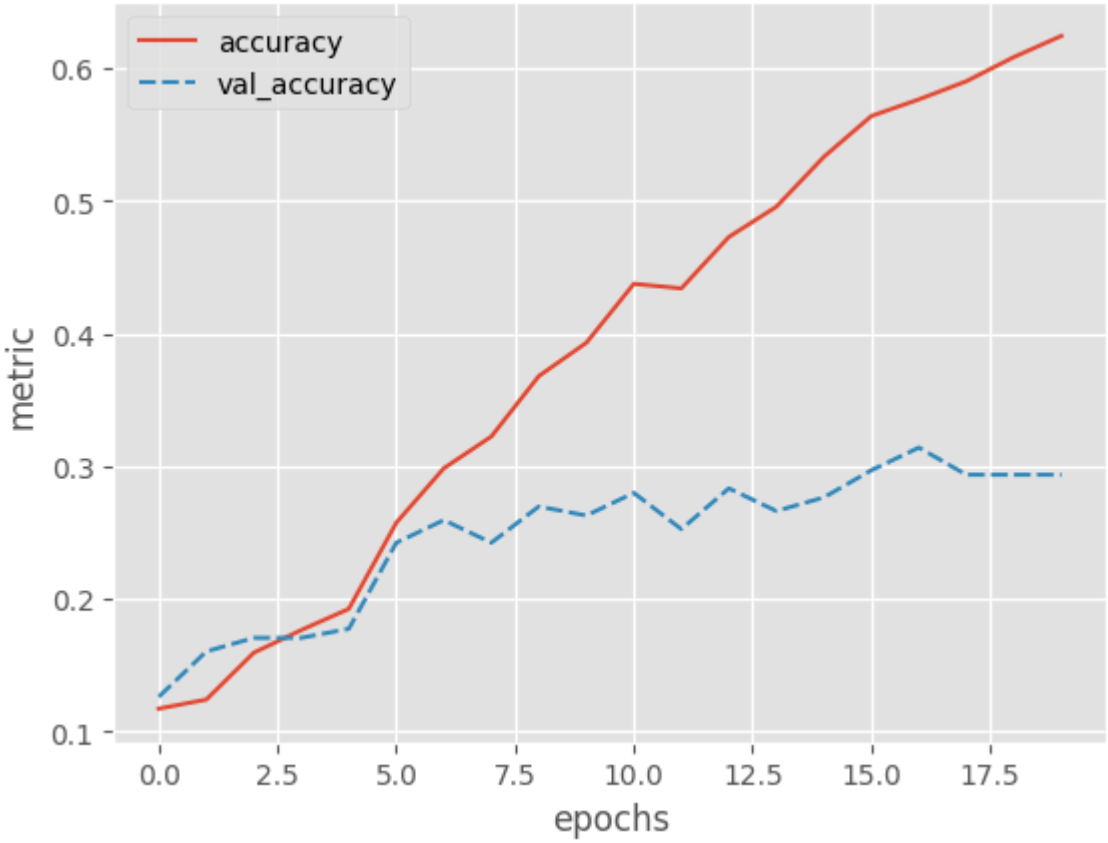
Trainable params: 137,060 (535.39 KB)

Non-trainable params: 0 (0.00 B)

In [93]:
```
compile_model(model_dropout)

history = model_dropout.fit(train_images, train_labels, epochs=20,
                    validation_data=(val_images, val_labels))
```

```
Epoch 1/20
28/28 ━━━━━━━━━━━━━━━━━━━━ 2s 43ms/step - accuracy: 0.1188 - loss: 2.3121
- val_accuracy: 0.1263 - val_loss: 2.2920
Epoch 2/20
28/28 ━━━━━━━━━━━━━━━━━━━━ 1s 41ms/step - accuracy: 0.1127 - loss: 2.2976
- val_accuracy: 0.1604 - val_loss: 2.2869
Epoch 3/20
28/28 ━━━━━━━━━━━━━━━━━━━━ 1s 41ms/step - accuracy: 0.1534 - loss: 2.2971
- val_accuracy: 0.1706 - val_loss: 2.2685
Epoch 4/20
28/28 ━━━━━━━━━━━━━━━━━━━━ 1s 40ms/step - accuracy: 0.1511 - loss: 2.2625
- val_accuracy: 0.1706 - val_loss: 2.2327
Epoch 5/20
28/28 ━━━━━━━━━━━━━━━━━━━━ 1s 41ms/step - accuracy: 0.1880 - loss: 2.1801
- val_accuracy: 0.1775 - val_loss: 2.2177
Epoch 6/20
28/28 ━━━━━━━━━━━━━━━━━━━━ 1s 42ms/step - accuracy: 0.2692 - loss: 2.0766
- val_accuracy: 0.2423 - val_loss: 2.1521
Epoch 7/20
28/28 ━━━━━━━━━━━━━━━━━━━━ 1s 40ms/step - accuracy: 0.3091 - loss: 1.9810
- val_accuracy: 0.2594 - val_loss: 2.1298
Epoch 8/20
28/28 ━━━━━━━━━━━━━━━━━━━━ 1s 41ms/step - accuracy: 0.3257 - loss: 1.8946
- val_accuracy: 0.2423 - val_loss: 2.1230
Epoch 9/20
28/28 ━━━━━━━━━━━━━━━━━━━━ 1s 41ms/step - accuracy: 0.3583 - loss: 1.8138
- val_accuracy: 0.2696 - val_loss: 2.1439
Epoch 10/20
28/28 ━━━━━━━━━━━━━━━━━━━━ 1s 42ms/step - accuracy: 0.3747 - loss: 1.7185
- val_accuracy: 0.2628 - val_loss: 2.2110
Epoch 11/20
28/28 ━━━━━━━━━━━━━━━━━━━━ 1s 41ms/step - accuracy: 0.4320 - loss: 1.6290
- val_accuracy: 0.2799 - val_loss: 2.2164
Epoch 12/20
28/28 ━━━━━━━━━━━━━━━━━━━━ 1s 40ms/step - accuracy: 0.4208 - loss: 1.5893
- val_accuracy: 0.2526 - val_loss: 2.2520
Epoch 13/20
28/28 ━━━━━━━━━━━━━━━━━━━━ 1s 41ms/step - accuracy: 0.4755 - loss: 1.4858
- val_accuracy: 0.2833 - val_loss: 2.2942
Epoch 14/20
28/28 ━━━━━━━━━━━━━━━━━━━━ 1s 41ms/step - accuracy: 0.5098 - loss: 1.3965
- val_accuracy: 0.2662 - val_loss: 2.3407
Epoch 15/20
28/28 ━━━━━━━━━━━━━━━━━━━━ 1s 42ms/step - accuracy: 0.5311 - loss: 1.3721
- val_accuracy: 0.2765 - val_loss: 2.4054
Epoch 16/20
28/28 ━━━━━━━━━━━━━━━━━━━━ 1s 42ms/step - accuracy: 0.5753 - loss: 1.2644
- val_accuracy: 0.2969 - val_loss: 2.3944
Epoch 17/20
28/28 ━━━━━━━━━━━━━━━━━━━━ 1s 41ms/step - accuracy: 0.5712 - loss: 1.2076
- val_accuracy: 0.3140 - val_loss: 2.4791
Epoch 18/20
28/28 ━━━━━━━━━━━━━━━━━━━━ 1s 41ms/step - accuracy: 0.5879 - loss: 1.1835
- val_accuracy: 0.2935 - val_loss: 2.5779
Epoch 19/20
28/28 ━━━━━━━━━━━━━━━━━━━━ 1s 42ms/step - accuracy: 0.6159 - loss: 1.0907
- val_accuracy: 0.2935 - val_loss: 2.6469
Epoch 20/20
28/28 ━━━━━━━━━━━━━━━━━━━━ 1s 41ms/step - accuracy: 0.6280 - loss: 1.0867
- val_accuracy: 0.2935 - val_loss: 2.6502
```

In [94]:
```
plot_history(history, ['accuracy', 'val_accuracy'])
```



In [95]:
```
def create_nn_with_hp(dropout_rate, n_layers):
    inputs = keras.Input(shape=train_images.shape[1:])
    x = inputs
    for layer in range(n_layers):
        x = keras.layers.Conv2D(50, (3, 3), activation='relu')(x)
        x = keras.layers.MaxPooling2D((2, 2))(x)
    x = keras.layers.Dropout(dropout_rate)(x)
    x = keras.layers.Flatten()(x)
    x = keras.layers.Dense(50, activation='relu')(x)
    outputs = keras.layers.Dense(10)(x)
    model = keras.Model(inputs=inputs, outputs=outputs, name="cifar_model
    return model
```

In [96]:
```
import keras_tuner

hp = keras_tuner.HyperParameters()

def build_model(hp):
    # Define values for hyperparameters to try out:
    n_layers = hp.Int("n_layers", min_value=1, max_value=2, step=1)
    dropout_rate = hp.Float("dropout_rate", min_value=0.2, max_value=0.8,

    model = create_nn_with_hp(dropout_rate, n_layers)
    compile_model(model)
    return model

tuner = keras_tuner.GridSearch(build_model, objective='val_loss')

tuner.search(train_images, train_labels, epochs=20,
             validation_data=(val_images, val_labels))
```

```
Trial 6 Complete [00h 00m 23s]
val_loss: 2.0955307483673096

Best val_loss So Far: 2.06892466545105
Total elapsed time: 00h 01m 53s
```

In [97]:
```
tuner.results_summary()
```

```
Results summary
Results in ./untitled_project
Showing 10 best trials
Objective(name="val_loss", direction="min")

Trial 0002 summary
Hyperparameters:
n_layers: 1
dropout_rate: 0.8
Score: 2.06892466545105

Trial 0000 summary
Hyperparameters:
n_layers: 1
dropout_rate: 0.2
Score: 2.092819929122925

Trial 0005 summary
Hyperparameters:
n_layers: 2
dropout_rate: 0.8
Score: 2.0955307483673096

Trial 0003 summary
Hyperparameters:
n_layers: 2
dropout_rate: 0.2
Score: 2.1084086894989014

Trial 0004 summary
Hyperparameters:
n_layers: 2
dropout_rate: 0.5
Score: 2.1336724758148193

Trial 0001 summary
Hyperparameters:
n_layers: 1
dropout_rate: 0.5
Score: 2.1546905040740967
```

In [98]:
```
model.save('cnn_model.keras')
```

In [99]:
```
!pwd
```

```
/Users/johannabayer/Documents
```

Transfer learning

In [103…
```
import pathlib
import numpy as np

DATA_FOLDER = pathlib.Path('data_dollar/') # change to location where you
train_images = np.load(DATA_FOLDER / 'train_images.npy')
val_images = np.load(DATA_FOLDER / 'test_images.npy')
train_labels = np.load(DATA_FOLDER / 'train_labels.npy')
val_labels = np.load(DATA_FOLDER / 'test_labels.npy')
```

In [107…
```
train_images = train_images / 255.0
val_images = val_images / 255.0
```

In [108…
```
# input tensor
inputs = keras.Input(train_images.shape[1:])
```

In [110…
```
# upscale layer
import tensorflow as tf
method = tf.image.ResizeMethod.BILINEAR
upscale = keras.layers.Lambda(
lambda x: tf.image.resize_with_pad(x, 160, 160, method=method))(inputs)
```

In [111…
```
base_model = keras.applications.DenseNet121(include_top=False,
                                             pooling='max',
                                             weights='imagenet',
                                             input_tensor=upscale,
                                             input_shape=(160,160,3),
                                             )
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-appl
ications/densenet/densenet121_weights_tf_dim_ordering_tf_kernels_notop.h5
```

```
---------------------------------------------------------------------------
SSLCertVerificationError                  Traceback (most recent call last)
File /Library/Frameworks/Python.framework/Versions/3.11/lib/python3.11/urllib/request.py:1348, in AbstractHTTPHandler.do_open(self, http_class, req, **http_conn_args)
   1347 try:
-> 1348     h.request(req.get_method(), req.selector, req.data, headers,
   1349               encode_chunked=req.has_header('Transfer-encoding'))
   1350 except OSError as err: # timeout error

File /Library/Frameworks/Python.framework/Versions/3.11/lib/python3.11/http/client.py:1286, in HTTPConnection.request(self, method, url, body, headers, encode_chunked)
   1285 """Send a complete request to the server."""
-> 1286 self._send_request(method, url, body, headers, encode_chunked)

File /Library/Frameworks/Python.framework/Versions/3.11/lib/python3.11/http/client.py:1332, in HTTPConnection._send_request(self, method, url, body, headers, encode_chunked)
   1331     body = _encode(body, 'body')
-> 1332 self.endheaders(body, encode_chunked=encode_chunked)

File /Library/Frameworks/Python.framework/Versions/3.11/lib/python3.11/http/client.py:1281, in HTTPConnection.endheaders(self, message_body, encode_chunked)
   1280     raise CannotSendHeader()
-> 1281 self._send_output(message_body, encode_chunked=encode_chunked)

File /Library/Frameworks/Python.framework/Versions/3.11/lib/python3.11/http/client.py:1041, in HTTPConnection._send_output(self, message_body, encode_chunked)
   1040 del self._buffer[:]
-> 1041 self.send(msg)
   1043 if message_body is not None:
   1044
   1045     # create a consistent interface to message_body

File /Library/Frameworks/Python.framework/Versions/3.11/lib/python3.11/http/client.py:979, in HTTPConnection.send(self, data)
    978 if self.auto_open:
--> 979     self.connect()
    980 else:

File /Library/Frameworks/Python.framework/Versions/3.11/lib/python3.11/http/client.py:1458, in HTTPSConnection.connect(self)
   1456     server_hostname = self.host
-> 1458 self.sock = self._context.wrap_socket(self.sock,
   1459                                       server_hostname=server_hostname)

File /Library/Frameworks/Python.framework/Versions/3.11/lib/python3.11/ssl.py:517, in SSLContext.wrap_socket(self, sock, server_side, do_handshake_on_connect, suppress_ragged_eofs, server_hostname, session)
    511 def wrap_socket(self, sock, server_side=False,
    512                 do_handshake_on_connect=True,
    513                 suppress_ragged_eofs=True,
    514                 server_hostname=None, session=None):
    515     # SSLSocket class handles server_hostname encoding before it c
```

```
alls
    516     # ctx._wrap_socket()
--> 517     return self.sslsocket_class._create(
    518         sock=sock,
    519         server_side=server_side,
    520         do_handshake_on_connect=do_handshake_on_connect,
    521         suppress_ragged_eofs=suppress_ragged_eofs,
    522         server_hostname=server_hostname,
    523         context=self,
    524         session=session
    525     )

File /Library/Frameworks/Python.framework/Versions/3.11/lib/python3.11/ssl.py:1075, in SSLSocket._create(cls, sock, server_side, do_handshake_on_connect, suppress_ragged_eofs, server_hostname, context, session)
   1074             raise ValueError("do_handshake_on_connect should not be specified for non-blocking sockets")
-> 1075         self.do_handshake()
   1076 except (OSError, ValueError):

File /Library/Frameworks/Python.framework/Versions/3.11/lib/python3.11/ssl.py:1346, in SSLSocket.do_handshake(self, block)
   1345         self.settimeout(None)
-> 1346     self._sslobj.do_handshake()
   1347 finally:

SSLCertVerificationError: [SSL: CERTIFICATE_VERIFY_FAILED] certificate verify failed: unable to get local issuer certificate (_ssl.c:1002)

During handling of the above exception, another exception occurred:

URLError                                  Traceback (most recent call last)
File ~/Documents/dl_workshop/lib/python3.11/site-packages/keras/src/utils/file_utils.py:291, in get_file(fname, origin, untar, md5_hash, file_hash, cache_subdir, hash_algorithm, extract, archive_format, cache_dir, force_download)
    290 try:
--> 291     urlretrieve(origin, fpath, DLProgbar())
    292 except urllib.error.HTTPError as e:

File /Library/Frameworks/Python.framework/Versions/3.11/lib/python3.11/urllib/request.py:241, in urlretrieve(url, filename, reporthook, data)
    239 url_type, path = _splittype(url)
--> 241 with contextlib.closing(urlopen(url, data)) as fp:
    242     headers = fp.info()

File /Library/Frameworks/Python.framework/Versions/3.11/lib/python3.11/urllib/request.py:216, in urlopen(url, data, timeout, cafile, capath, cadefault, context)
    215     opener = _opener
--> 216 return opener.open(url, data, timeout)

File /Library/Frameworks/Python.framework/Versions/3.11/lib/python3.11/urllib/request.py:519, in OpenerDirector.open(self, fullurl, data, timeout)
    518 sys.audit('urllib.Request', req.full_url, req.data, req.headers, req.get_method())
--> 519 response = self._open(req, data)
    521 # post-process response
```

```
File /Library/Frameworks/Python.framework/Versions/3.11/lib/python3.11/url
lib/request.py:536, in OpenerDirector._open(self, req, data)
    535 protocol = req.type
--> 536 result = self._call_chain(self.handle_open, protocol, protocol +
    537                           '_open', req)
    538 if result:

File /Library/Frameworks/Python.framework/Versions/3.11/lib/python3.11/url
lib/request.py:496, in OpenerDirector._call_chain(self, chain, kind, meth_
name, *args)
    495 func = getattr(handler, meth_name)
--> 496 result = func(*args)
    497 if result is not None:

File /Library/Frameworks/Python.framework/Versions/3.11/lib/python3.11/url
lib/request.py:1391, in HTTPSHandler.https_open(self, req)
   1390 def https_open(self, req):
-> 1391     return self.do_open(http.client.HTTPSConnection, req,
   1392         context=self._context, check_hostname=self._check_hostnam
e)

File /Library/Frameworks/Python.framework/Versions/3.11/lib/python3.11/url
lib/request.py:1351, in AbstractHTTPHandler.do_open(self, http_class, req,
**http_conn_args)
   1350 except OSError as err: # timeout error
-> 1351     raise URLError(err)
   1352 r = h.getresponse()

URLError: <urlopen error [SSL: CERTIFICATE_VERIFY_FAILED] certificate veri
fy failed: unable to get local issuer certificate (_ssl.c:1002)>

During handling of the above exception, another exception occurred:

Exception                                 Traceback (most recent call las
t)
Cell In[111], line 1
----> 1 base_model = keras.applications.DenseNet121(include_top=False,
      2                                             pooling='max',
      3                                             weights='imagenet',
      4                                             input_tensor=upscale,
      5                                             input_shape=(160,160,
3),
      6                                             )

File ~/Documents/dl_workshop/lib/python3.11/site-packages/keras/src/applic
ations/densenet.py:338, in DenseNet121(include_top, weights, input_tensor,
input_shape, pooling, classes, classifier_activation, name)
    321 @keras_export(
    322     [
    323         "keras.applications.densenet.DenseNet121",
   (...)
    335     name="densenet121",
    336 ):
    337     """Instantiates the Densenet121 architecture."""
--> 338     return DenseNet(
    339         [6, 12, 24, 16],
    340         include_top,
    341         weights,
    342         input_tensor,
    343         input_shape,
```

```
    344         pooling,
    345         classes,
    346         classifier_activation,
    347         name=name,
    348     )

File ~/Documents/dl_workshop/lib/python3.11/site-packages/keras/src/applic
ations/densenet.py:294, in DenseNet(blocks, include_top, weights, input_te
nsor, input_shape, pooling, classes, classifier_activation, name)
    292 else:
    293     if blocks == [6, 12, 24, 16]:
--> 294         weights_path = file_utils.get_file(
    295             "densenet121_weights_tf_dim_ordering_tf_kernels_notop.
h5",
    296             DENSENET121_WEIGHT_PATH_NO_TOP,
    297             cache_subdir="models",
    298             file_hash="30ee3e1110167f948a6b9946edeeb738",
    299         )
    300     elif blocks == [6, 12, 32, 32]:
    301         weights_path = file_utils.get_file(
    302             "densenet169_weights_tf_dim_ordering_tf_kernels_notop.
h5",
    303             DENSENET169_WEIGHT_PATH_NO_TOP,
    304             cache_subdir="models",
    305             file_hash="b8c4d4c20dd625c148057b9ff1c1176b",
    306         )

File ~/Documents/dl_workshop/lib/python3.11/site-packages/keras/src/utils/
file_utils.py:295, in get_file(fname, origin, untar, md5_hash, file_hash,
cache_subdir, hash_algorithm, extract, archive_format, cache_dir, force_do
wnload)
    293         raise Exception(error_msg.format(origin, e.code, e.msg))
    294     except urllib.error.URLError as e:
--> 295         raise Exception(error_msg.format(origin, e.errno, e.reaso
n))
    296 except (Exception, KeyboardInterrupt):
    297     if os.path.exists(fpath):

Exception: URL fetch failure on https://storage.googleapis.com/tensorflow/
keras-applications/densenet/densenet121_weights_tf_dim_ordering_tf_kernels
_notop.h5: None -- [SSL: CERTIFICATE_VERIFY_FAILED] certificate verify fai
led: unable to get local issuer certificate (_ssl.c:1002)
```

In [126…
```
from tensorflow.keras.applications import DenseNet121

base_model = keras.applications.DenseNet121(include_top=False,
                                            pooling='max',
                                            weights='densenet121_weights_
                                            input_tensor=upscale,
                                            input_shape=(160,160,3),
                                            )
```

In [127…
```
base_model.trainable = False
```

In [128…
```
out = base_model.output
out = keras.layers.Flatten()(out)
out = keras.layers.BatchNormalization()(out)
out = keras.layers.Dense(50, activation='relu')(out)
```

```
out = keras.layers.Dropout(0.5)(out)
out = keras.layers.Dense(10)(out)
```

In [129…
```
model = keras.models.Model(inputs=inputs, outputs=out)
```

In [130…
```
model.compile(optimizer='adam',
              loss=keras.losses.SparseCategoricalCrossentropy(from_logits
              metrics=['accuracy'])
```

In [131…
```
early_stopper = keras.callbacks.EarlyStopping(monitor='val_accuracy',
                                  patience=5)
```

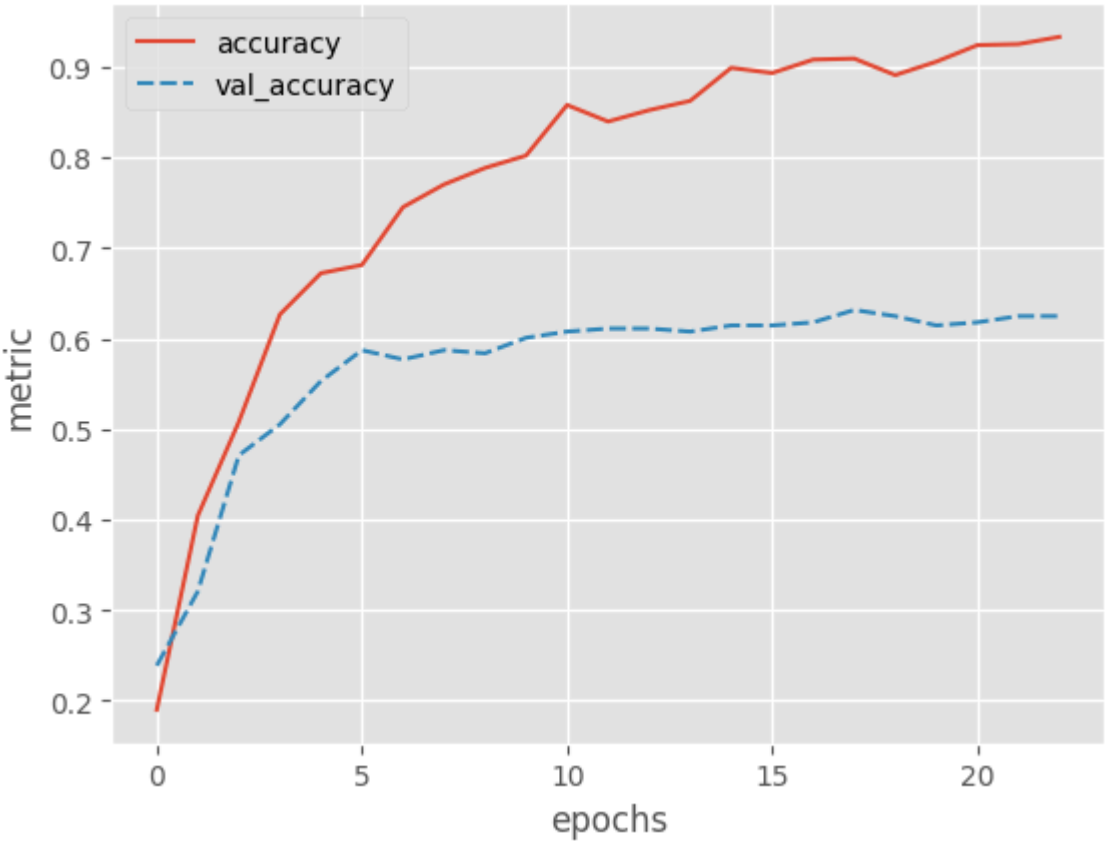
In [132…
```
history = model.fit(x=train_images,
                    y=train_labels,
                    batch_size=32,
                    epochs=30,
                    callbacks=[early_stopper],
                    validation_data=(val_images, val_labels))
```

```
Epoch 1/30
28/28 ━━━━━━━━━━━━━━━━━━━━ 20s 600ms/step - accuracy: 0.1378 - loss: 2.783
9 - val_accuracy: 0.2389 - val_loss: 2.3437
Epoch 2/30
28/28 ━━━━━━━━━━━━━━━━━━━━ 16s 563ms/step - accuracy: 0.4049 - loss: 1.735
3 - val_accuracy: 0.3208 - val_loss: 1.8731
Epoch 3/30
28/28 ━━━━━━━━━━━━━━━━━━━━ 16s 567ms/step - accuracy: 0.4977 - loss: 1.430
9 - val_accuracy: 0.4710 - val_loss: 1.6147
Epoch 4/30
28/28 ━━━━━━━━━━━━━━━━━━━━ 16s 573ms/step - accuracy: 0.6266 - loss: 1.144
4 - val_accuracy: 0.5051 - val_loss: 1.5220
Epoch 5/30
28/28 ━━━━━━━━━━━━━━━━━━━━ 16s 569ms/step - accuracy: 0.6939 - loss: 1.012
3 - val_accuracy: 0.5529 - val_loss: 1.4306
Epoch 6/30
28/28 ━━━━━━━━━━━━━━━━━━━━ 16s 569ms/step - accuracy: 0.6699 - loss: 0.956
5 - val_accuracy: 0.5870 - val_loss: 1.3628
Epoch 7/30
28/28 ━━━━━━━━━━━━━━━━━━━━ 16s 566ms/step - accuracy: 0.7386 - loss: 0.815
4 - val_accuracy: 0.5768 - val_loss: 1.3456
Epoch 8/30
28/28 ━━━━━━━━━━━━━━━━━━━━ 16s 558ms/step - accuracy: 0.7845 - loss: 0.720
8 - val_accuracy: 0.5870 - val_loss: 1.3135
Epoch 9/30
28/28 ━━━━━━━━━━━━━━━━━━━━ 15s 557ms/step - accuracy: 0.7973 - loss: 0.605
7 - val_accuracy: 0.5836 - val_loss: 1.2887
Epoch 10/30
28/28 ━━━━━━━━━━━━━━━━━━━━ 16s 578ms/step - accuracy: 0.7834 - loss: 0.631
3 - val_accuracy: 0.6007 - val_loss: 1.2540
Epoch 11/30
28/28 ━━━━━━━━━━━━━━━━━━━━ 16s 560ms/step - accuracy: 0.8585 - loss: 0.507
1 - val_accuracy: 0.6075 - val_loss: 1.2561
Epoch 12/30
28/28 ━━━━━━━━━━━━━━━━━━━━ 15s 556ms/step - accuracy: 0.8255 - loss: 0.528
6 - val_accuracy: 0.6109 - val_loss: 1.2478
Epoch 13/30
28/28 ━━━━━━━━━━━━━━━━━━━━ 16s 560ms/step - accuracy: 0.8440 - loss: 0.471
3 - val_accuracy: 0.6109 - val_loss: 1.2518
Epoch 14/30
28/28 ━━━━━━━━━━━━━━━━━━━━ 15s 557ms/step - accuracy: 0.8579 - loss: 0.455
0 - val_accuracy: 0.6075 - val_loss: 1.2616
Epoch 15/30
28/28 ━━━━━━━━━━━━━━━━━━━━ 16s 578ms/step - accuracy: 0.9043 - loss: 0.375
2 - val_accuracy: 0.6143 - val_loss: 1.2715
Epoch 16/30
28/28 ━━━━━━━━━━━━━━━━━━━━ 16s 568ms/step - accuracy: 0.9040 - loss: 0.344
5 - val_accuracy: 0.6143 - val_loss: 1.2661
Epoch 17/30
28/28 ━━━━━━━━━━━━━━━━━━━━ 16s 559ms/step - accuracy: 0.9009 - loss: 0.350
0 - val_accuracy: 0.6177 - val_loss: 1.2882
Epoch 18/30
28/28 ━━━━━━━━━━━━━━━━━━━━ 15s 551ms/step - accuracy: 0.9036 - loss: 0.331
2 - val_accuracy: 0.6314 - val_loss: 1.2811
Epoch 19/30
28/28 ━━━━━━━━━━━━━━━━━━━━ 16s 563ms/step - accuracy: 0.8908 - loss: 0.325
6 - val_accuracy: 0.6246 - val_loss: 1.2965
Epoch 20/30
28/28 ━━━━━━━━━━━━━━━━━━━━ 16s 563ms/step - accuracy: 0.9209 - loss: 0.300
9 - val_accuracy: 0.6143 - val_loss: 1.2913
```

```
Epoch 21/30
28/28 ━━━━━━━━━━━━━━━━━━━━ 16s 567ms/step - accuracy: 0.9221 - loss: 0.264
2 - val_accuracy: 0.6177 - val_loss: 1.3150
Epoch 22/30
28/28 ━━━━━━━━━━━━━━━━━━━━ 16s 561ms/step - accuracy: 0.9423 - loss: 0.244
4 - val_accuracy: 0.6246 - val_loss: 1.3183
Epoch 23/30
28/28 ━━━━━━━━━━━━━━━━━━━━ 16s 572ms/step - accuracy: 0.9389 - loss: 0.256
1 - val_accuracy: 0.6246 - val_loss: 1.3210
```

In [133…

```python
def plot_history(history, metrics):
    """
    Plot the training history

    Args:
        history (keras History object that is returned by model.fit())
        metrics(str, list): Metric or a list of metrics to plot
    """
    history_df = pd.DataFrame.from_dict(history.history)
    sns.lineplot(data=history_df[metrics])
    plt.xlabel("epochs")
    plt.ylabel("metric")

plot_history(history, ['accuracy', 'val_accuracy'])
```



In [ ]: