CODE:

```
import os
for dirname, _, filenames in os.walk('/kaggle/input'):
for filename in filenames:
print(os.path.join(dirname, filename))
!pip install -q -U keras-tuner
import numpy as np
import matplotlib.pyplot as plt
import random
# ----------------------------------
import tensorflow as tf
import keras
from tensorflow import keras
import keras_tuner as kt
fashion_mnist = keras.datasets.fashion_mnist
(train_images, train_labels), (test_images, test_labels) =
fashion_mnist.load_data()
train_images   = train_images / 255.0
test_images = test_images / 255.0
plt.figure(figsize=(10,10))
for i in range(16):
```

```
    plt.subplot(4,4,i+1)
    plt.xticks([])
    plt.yticks([])
    plt.grid(False)
    plt.imshow(train_images[i], cmap=plt.cm.binary)
    plt.title("Image label is: {}".format(train_labels[i]))
plt.show()
x_train = train_images.reshape(-1,28,28,1)
x_test = test_images.reshape(-1,28,28,1)
early_stop = keras.callbacks.EarlyStopping(monitor='val_loss',
patience=3)
def build_model(hp):
    model = keras.Sequential([
        # First conv_block
    keras.layers.Conv2D(
        filters = hp.Choice('conv_1_filter', values=[16, 32, 64, 128]),
        kernel_size=hp.Choice('conv_1_kernel', values = [3,4]),
        activation='relu',
        input_shape=(28,28,1)),
    keras.layers.MaxPooling2D((2,2)),
```

```
    # Second conv_block
    keras.layers.Conv2D(
        filters = hp.Choice('conv_2_filter', values=[16, 32, 64, 128]),
        kernel_size=hp.Choice('conv_2_kernel', values = [3,4]),
        activation='relu'),
    keras.layers.MaxPooling2D((2,2)),

    # --------------------------------
    keras.layers.Flatten(),
    keras.layers.Dense(units = hp.Choice('units', values=[16, 32, 64, 128,
256]),
                        activation='relu'),
    keras.layers.Dropout(hp.Float('dropout', 0, 0.5, step=0.1,
default=0.5)),

    # --------------------------------
     keras.layers.Dense(10)
    ])


model.compile(optimizer=keras.optimizers.Adam(hp.Choice('learning_r
ate',
```

```
values=[1e-1, 1e-2, 1e-3, 1e-4])),

loss=keras.losses.SparseCategoricalCrossentropy(from_logits=True),
                   metrics=['accuracy'])
      return model
tuner = kt.Hyperband(build_model,
                             objective="val_accuracy",
                             max_epochs=5,
                             factor=3,
                             hyperband_iterations=3)
tuner.search_space_summary()
tuner.search(x_train,train_labels, epochs=3, validation_split=0.2)
best_hps = best_hps=tuner.get_best_hyperparameters(num_trials=1)[0]
print(f"""conv_1_filter is {best_hps.get('conv_1_filter')}""")
print(f"""conv_1_kernel is {best_hps.get('conv_1_kernel')}""")
print(f"""conv_2_filter is {best_hps.get('conv_2_filter')}""")
print(f"""conv_2_kernel is {best_hps.get('conv_2_kernel')}""")
print("----------------------------------------------")
print(f"""units is {best_hps.get('units')}""")
print(f"""learning_rate is {best_hps.get('learning_rate')}""")
```

```
print(f"""dropout is {best_hps.get('dropout')}""")

model = tuner.hypermodel.build(best_hps)

history = model.fit(x_train, train_labels,

                    epochs=50, validation_split=0.2)

val_acc_per_epoch = history.history['val_accuracy']

best_epoch = val_acc_per_epoch.index(max(val_acc_per_epoch)) + 1

print('Best epoch: %d' % (best_epoch,))

hypermodel = tuner.hypermodel.build(best_hps)

history = hypermodel.fit(x_train, train_labels,

                         epochs=best_epoch,

                         validation_split=0.2,

                         callbacks=[early_stop])

hypermodel.summary()

keras.utils.plot_model(hypermodel, show_shapes=True)

successive_outputs = [layer.output for layer in hypermodel.layers[1:]]

visualization_model = keras.models.Model(inputs = hypermodel.input,
outputs = successive_outputs)

index = 20

plt.imshow(train_images[index], cmap=plt.cm.binary)


x = train_images[index]
```

```
x = x.reshape((1,) + x.shape)

x /= 255

successive_feature_maps = visualization_model.predict(x)

layer_names = [layer.name for layer in hypermodel.layers[1:]]


for layer_name, feature_map in zip(layer_names,
successive_feature_maps):

    if len(feature_map.shape) == 4:

        n_features = feature_map.shape[-1]

        size = feature_map.shape[1]

        display_grid = np.zeros((size, size * n_features))

        for i in range(n_features):

            x = feature_map[0, :, :, i]

            x -= x.mean()

            x /= x.std()

            x *= 64

            x += 128

            x = np.clip(x, 0, 255).astype('uint8')

            display_grid[:, i * size : (i + 1) * size] = x

            scale = 20. / n_features

    plt.figure(figsize=(scale * n_features, scale))
```

```
    plt.title(layer_name)
    plt.grid(False)
    plt.imshow(display_grid, aspect='auto', cmap='viridis')
    eval_result = hypermodel.evaluate(x_test, test_labels)
print("[test loss, test accuracy]:", eval_result)
pred = hypermodel.predict(x_test)
print("Prediction is -> {}".format(pred[12]))
print("Actual value is -> {}".format(test_labels[12]))
print("The highest value for label is {}".format(np.argmax(pred[12])))
import matplotlib.pyplot as plt
%matplotlib inline
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']
loss = history.history['loss']
val_loss = history.history['val_loss']
epochs = range(len(acc))
plt.plot(epochs, acc, 'bo', label='Training acc')
plt.plot(epochs, val_acc, 'b', label='Validation acc')
plt.title('Training and validation accuracy')
plt.legend()
plt.figure()
```

```
plt.plot(epochs, loss, 'bo', label='Training loss')

plt.plot(epochs, val_loss, 'b', label='Validation loss')

plt.title('Training and validation loss')

plt.legend()

plt.show()
```

OUTPUT:

```
Downloading data from
https://storage.googleapis.com/tensorflow/tf-keras-datasets/train-labels-idx1-ubyte.gz

29515/29515 [==============================] - 0s 0us/step

Downloading data from
https://storage.googleapis.com/tensorflow/tf-keras-datasets/train-images-idx3-ubyte.gz

26421880/26421880 [==============================] - 1s 0us/step

Downloading data from
https://storage.googleapis.com/tensorflow/tf-keras-datasets/t10k-labels-idx1-ubyte.gz

5148/5148 [==============================] - 0s 0us/step

Downloading data from
https://storage.googleapis.com/tensorflow/tf-keras-datasets/t10k-images-
```

idx3-ubyte.gz

4422102/4422102 [==============================] - 1s 0us/step

The labels are an array of integers, ranging from 0 to 9.

Label	Class
0	T-shirt/top
1	Trouser
2	Pullover
3	Dress
4	Coat
5	Sandal
6	Shirt
7	Sneaker
8	Bag
9	Ankle boot

Search space summary

Default search space size: 7

conv_1_filter (Choice)

{'default': 16, 'conditions': [], 'values': [16, 32, 64, 128], 'ordered': True}

conv_1_kernel (Choice)

{'default': 3, 'conditions': [], 'values': [3, 4], 'ordered': True}

conv_2_filter (Choice)

{'default': 16, 'conditions': [], 'values': [16, 32, 64, 128], 'ordered': True}

conv_2_kernel (Choice)

{'default': 3, 'conditions': [], 'values': [3, 4], 'ordered': True}

units (Choice)

{'default': 16, 'conditions': [], 'values': [16, 32, 64, 128, 256], 'ordered': True}

dropout (Float)

{'default': 0.5, 'conditions': [], 'min_value': 0.0, 'max_value': 0.5, 'step': 0.1, 'sampling': 'linear'}

learning_rate (Choice)

{'default': 0.1, 'conditions': [], 'values': [0.1, 0.01, 0.001, 0.0001], 'ordered': True}

Trial 30 Complete [00h 06m 40s]

val_accuracy: 0.8593333363533302

Best val_accuracy So Far: 0.9104166626930237

Total elapsed time: 02h 08m 04s

conv_1_filter is 16

conv_1_kernel is 3

conv_2_filter is 128

conv_2_kernel is 3

----

units is 64

learning_rate is 0.001

dropout is 0.0

Epoch 1/50

1500/1500 [==============================] - 40s 26ms/step - loss: 0.4769 - accuracy: 0.8289 - val_loss: 0.3559 - val_accuracy: 0.8709

Epoch 2/50

1500/1500 [==============================] - 39s 26ms/step - loss: 0.3170 - accuracy: 0.8853 - val_loss: 0.3248 - val_accuracy: 0.8827

Epoch 3/50

1500/1500 [==============================] - 41s 27ms/step - loss: 0.2709 - accuracy: 0.9019 - val_loss: 0.2815 - val_accuracy: 0.8982

Epoch 4/50

1500/1500 [==============================] - 39s 26ms/step - loss: 0.2377 - accuracy: 0.9125 - val_loss: 0.2654 - val_accuracy: 0.9028

Epoch 5/50

1500/1500 [==============================] - 39s 26ms/step - loss: 0.2114 - accuracy: 0.9222 - val_loss: 0.2509 - val_accuracy: 0.9133

Epoch 6/50

1500/1500 [==============================] - 38s 26ms/step - loss: 0.1844 - accuracy: 0.9326 - val_loss: 0.2682 - val_accuracy: 0.9069

Epoch 7/50

1500/1500 [==============================] - 38s 25ms/step -

loss: 0.1647 - accuracy: 0.9386 - val_loss: 0.2582 - val_accuracy: 0.9123

Epoch 8/50

1500/1500 [==============================] - 38s 25ms/step - loss: 0.1443 - accuracy: 0.9460 - val_loss: 0.2756 - val_accuracy: 0.9079

Epoch 9/50

1500/1500 [==============================] - 38s 25ms/step - loss: 0.1280 - accuracy: 0.9525 - val_loss: 0.2721 - val_accuracy: 0.9116

Epoch 10/50

1500/1500 [==============================] - 39s 26ms/step - loss: 0.1144 - accuracy: 0.9567 - val_loss: 0.2813 - val_accuracy: 0.9159

Epoch 11/50

1500/1500 [==============================] - 40s 26ms/step - loss: 0.1016 - accuracy: 0.9616 - val_loss: 0.2899 - val_accuracy: 0.9181

Epoch 12/50

1500/1500 [==============================] - 41s 27ms/step - loss: 0.0871 - accuracy: 0.9679 - val_loss: 0.3115 - val_accuracy: 0.9097

Epoch 13/50

1500/1500 [==============================] - 38s 26ms/step - loss: 0.0768 - accuracy: 0.9718 - val_loss: 0.3440 - val_accuracy: 0.9136

Epoch 14/50

1500/1500 [==============================] - 38s 25ms/step - loss: 0.0710 - accuracy: 0.9734 - val_loss: 0.3411 - val_accuracy: 0.9111

Epoch 15/50

1500/1500 [==============================] - 41s 27ms/step - loss: 0.0620 - accuracy: 0.9770 - val_loss: 0.3829 - val_accuracy: 0.9035

Epoch 16/50

1500/1500 [==============================] - 38s 26ms/step - loss: 0.0565 - accuracy: 0.9783 - val_loss: 0.3907 - val_accuracy: 0.9069

Epoch 17/50

1500/1500 [==============================] - 39s 26ms/step - loss: 0.0490 - accuracy: 0.9809 - val_loss: 0.4521 - val_accuracy: 0.9103

Epoch 18/50

1500/1500 [==============================] - 39s 26ms/step - loss: 0.0477 - accuracy: 0.9815 - val_loss: 0.4447 - val_accuracy: 0.9083

Epoch 19/50

1500/1500 [==============================] - 40s 27ms/step - loss: 0.0414 - accuracy: 0.9851 - val_loss: 0.4517 - val_accuracy: 0.9055

Epoch 20/50

1500/1500 [==============================] - 38s 25ms/step - loss: 0.0425 - accuracy: 0.9842 - val_loss: 0.4762 - val_accuracy: 0.9062

Epoch 21/50

1500/1500 [==============================] - 38s 25ms/step - loss: 0.0370 - accuracy: 0.9860 - val_loss: 0.4721 - val_accuracy: 0.9093

Epoch 22/50

1500/1500 [==============================] - 38s 25ms/step - loss: 0.0347 - accuracy: 0.9874 - val_loss: 0.4976 - val_accuracy: 0.9075

Epoch 23/50

1500/1500 [==============================] - 38s 25ms/step - loss: 0.0345 - accuracy: 0.9873 - val_loss: 0.5230 - val_accuracy: 0.9014

Epoch 24/50

1500/1500 [==============================] - 38s 25ms/step - loss: 0.0307 - accuracy: 0.9891 - val_loss: 0.5560 - val_accuracy: 0.9098

Epoch 25/50

1500/1500 [==============================] - 39s 26ms/step - loss: 0.0317 - accuracy: 0.9887 - val_loss: 0.5262 - val_accuracy: 0.9107

Epoch 26/50

1500/1500 [==============================] - 41s 27ms/step - loss: 0.0300 - accuracy: 0.9896 - val_loss: 0.5346 - val_accuracy: 0.9078

Epoch 27/50

1500/1500 [==============================] - 38s 25ms/step - loss: 0.0261 - accuracy: 0.9907 - val_loss: 0.5753 - val_accuracy: 0.9118

Epoch 28/50

1500/1500 [==============================] - 38s 25ms/step - loss: 0.0272 - accuracy: 0.9903 - val_loss: 0.5749 - val_accuracy: 0.9072

Epoch 29/50

1500/1500 [==============================] - 38s 25ms/step - loss: 0.0255 - accuracy: 0.9906 - val_loss: 0.7044 - val_accuracy: 0.8963

Epoch 30/50

1500/1500 [==============================] - 38s 25ms/step -

loss: 0.0246 - accuracy: 0.9915 - val_loss: 0.6231 - val_accuracy: 0.9089

Epoch 31/50

1500/1500 [==============================] - 40s 26ms/step - loss: 0.0241 - accuracy: 0.9913 - val_loss: 0.6572 - val_accuracy: 0.9055

Epoch 32/50

1500/1500 [==============================] - 39s 26ms/step - loss: 0.0233 - accuracy: 0.9921 - val_loss: 0.6707 - val_accuracy: 0.9085

Epoch 33/50

1500/1500 [==============================] - 38s 25ms/step - loss: 0.0272 - accuracy: 0.9911 - val_loss: 0.6709 - val_accuracy: 0.9000

Epoch 34/50

1500/1500 [==============================] - 41s 28ms/step - loss: 0.0193 - accuracy: 0.9931 - val_loss: 0.6756 - val_accuracy: 0.9085

Epoch 35/50

1500/1500 [==============================] - 39s 26ms/step - loss: 0.0238 - accuracy: 0.9916 - val_loss: 0.6779 - val_accuracy: 0.9075

Epoch 36/50

1500/1500 [==============================] - 39s 26ms/step - loss: 0.0211 - accuracy: 0.9928 - val_loss: 0.7816 - val_accuracy: 0.9038

Epoch 37/50

1500/1500 [==============================] - 39s 26ms/step - loss: 0.0212 - accuracy: 0.9925 - val_loss: 0.7314 - val_accuracy: 0.8977

Epoch 38/50

1500/1500 [==============================] - 43s 28ms/step - loss: 0.0165 - accuracy: 0.9944 - val_loss: 0.8190 - val_accuracy: 0.9055

Epoch 39/50

1500/1500 [==============================] - 40s 27ms/step - loss: 0.0225 - accuracy: 0.9921 - val_loss: 0.7511 - val_accuracy: 0.9041

Epoch 40/50

1500/1500 [==============================] - 38s 25ms/step - loss: 0.0189 - accuracy: 0.9934 - val_loss: 0.7485 - val_accuracy: 0.9068

Epoch 41/50

1500/1500 [==============================] - 39s 26ms/step - loss: 0.0184 - accuracy: 0.9936 - val_loss: 0.7445 - val_accuracy: 0.9091

Epoch 42/50

1500/1500 [==============================] - 38s 26ms/step - loss: 0.0191 - accuracy: 0.9939 - val_loss: 0.8607 - val_accuracy: 0.9057

Epoch 43/50

1500/1500 [==============================] - 38s 26ms/step - loss: 0.0203 - accuracy: 0.9933 - val_loss: 0.7747 - val_accuracy: 0.9110

Epoch 44/50

1500/1500 [==============================] - 42s 28ms/step - loss: 0.0183 - accuracy: 0.9938 - val_loss: 0.7727 - val_accuracy: 0.9064

Epoch 45/50

1500/1500 [==============================] - 38s 25ms/step - loss: 0.0187 - accuracy: 0.9937 - val_loss: 0.8216 - val_accuracy: 0.9110

Epoch 46/50

1500/1500 [==============================] - 39s 26ms/step - loss: 0.0182 - accuracy: 0.9938 - val_loss: 0.8518 - val_accuracy: 0.9078

Epoch 47/50

1500/1500 [==============================] - 38s 26ms/step - loss: 0.0158 - accuracy: 0.9943 - val_loss: 0.8267 - val_accuracy: 0.9074

Epoch 48/50

1500/1500 [==============================] - 38s 25ms/step - loss: 0.0149 - accuracy: 0.9949 - val_loss: 0.8533 - val_accuracy: 0.9097

Epoch 49/50

1500/1500 [==============================] - 39s 26ms/step - loss: 0.0192 - accuracy: 0.9936 - val_loss: 0.8208 - val_accuracy: 0.9079

Epoch 50/50

1500/1500 [==============================] - 39s 26ms/step - loss: 0.0185 - accuracy: 0.9939 - val_loss: 0.8737 - val_accuracy: 0.9066

Best epoch: 11

Epoch 1/11

1500/1500 [==============================] - 41s 26ms/step - loss: 0.4922 - accuracy: 0.8225 - val_loss: 0.3639 - val_accuracy: 0.8692

Epoch 2/11

1500/1500 [==============================] - 39s 26ms/step - loss: 0.3275 - accuracy: 0.8814 - val_loss: 0.3058 - val_accuracy: 0.8913

Epoch 3/11

1500/1500 [==============================] - 39s 26ms/step - loss: 0.2793 - accuracy: 0.8984 - val_loss: 0.2890 - val_accuracy: 0.8947

Epoch 4/11

1500/1500 [==============================] - 41s 28ms/step - loss: 0.2449 - accuracy: 0.9101 - val_loss: 0.2698 - val_accuracy: 0.9010

Epoch 5/11

1500/1500 [==============================] - 39s 26ms/step - loss: 0.2187 - accuracy: 0.9200 - val_loss: 0.2633 - val_accuracy: 0.9062

Epoch 6/11

1500/1500 [==============================] - 42s 28ms/step - loss: 0.1963 - accuracy: 0.9278 - val_loss: 0.2606 - val_accuracy: 0.9056

Epoch 7/11

1500/1500 [==============================] - 41s 27ms/step - loss: 0.1765 - accuracy: 0.9351 - val_loss: 0.2498 - val_accuracy: 0.9121

Epoch 8/11

1500/1500 [==============================] - 39s 26ms/step - loss: 0.1572 - accuracy: 0.9412 - val_loss: 0.2565 - val_accuracy: 0.9112

Epoch 9/11

1500/1500 [==============================] - 39s 26ms/step - loss: 0.1426 - accuracy: 0.9468 - val_loss: 0.2706 - val_accuracy: 0.9080

Epoch 10/11

1500/1500 [==============================] - 39s 26ms/step - loss: 0.1252 - accuracy: 0.9534 - val_loss: 0.2631 - val_accuracy: 0.9161

```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_4 (Conv2D)           (None, 26, 26, 16)        160

 max_pooling2d_4 (MaxPoolin  (None, 13, 13, 16)        0
 g2D)

 conv2d_5 (Conv2D)           (None, 11, 11, 128)       18560

 max_pooling2d_5 (MaxPoolin  (None, 5, 5, 128)         0
 g2D)

 flatten_2 (Flatten)         (None, 3200)              0

 dense_4 (Dense)             (None, 64)                204864

 dropout_2 (Dropout)         (None, 64)                0
```

dense_5 (Dense)   (None, 10)   650

===

Total params: 224234 (875.91 KB)

Trainable params: 224234 (875.91 KB)

Non-trainable params: 0 (0.00 Byte)

____

1/1 [==============================] - 0s 129ms/step

<ipython-input-10-bac1e203d4b6>:111: RuntimeWarning: invalid value encountered in divide

  x /= x.std()

<ipython-input-10-bac1e203d4b6>:114: RuntimeWarning: invalid value encountered in cast

  x = np.clip(x, 0, 255).astype('uint8')

313/313 [==============================] - 3s 8ms/step - loss: 0.2634 - accuracy: 0.9144

313/313 [==============================] - 3s 11ms/step - loss: 0.2634 - accuracy: 0.9144

313/313 [==============================] - 3s 8ms/step - loss: 0.2634 - accuracy: 0.9144

313/313 [==============================] - 3s 8ms/step - loss: 0.2634 - accuracy: 0.9144

313/313 [==============================] - 2s 8ms/step - loss: 0.2634 - accuracy: 0.9144

313/313 [==============================] - 3s 11ms/step - loss: 0.2634 - accuracy: 0.9144

313/313 [==============================] - 2s 8ms/step - loss: 0.2634 - accuracy: 0.9144

[test loss, test accuracy]: [0.26343563199043274, 0.9143999814987183]

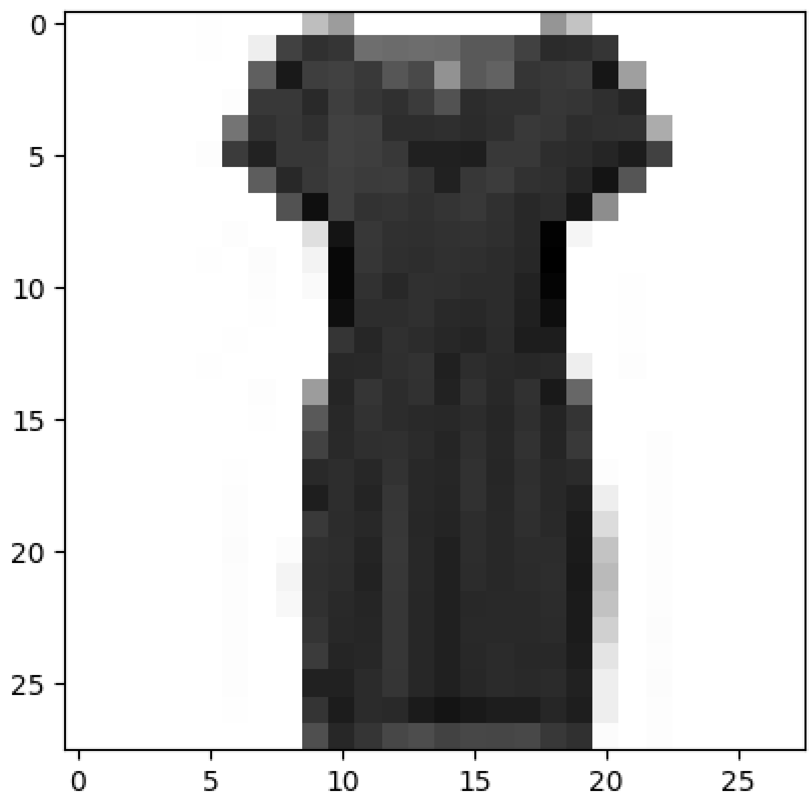
313/313 [==============================] - 2s 8ms/step

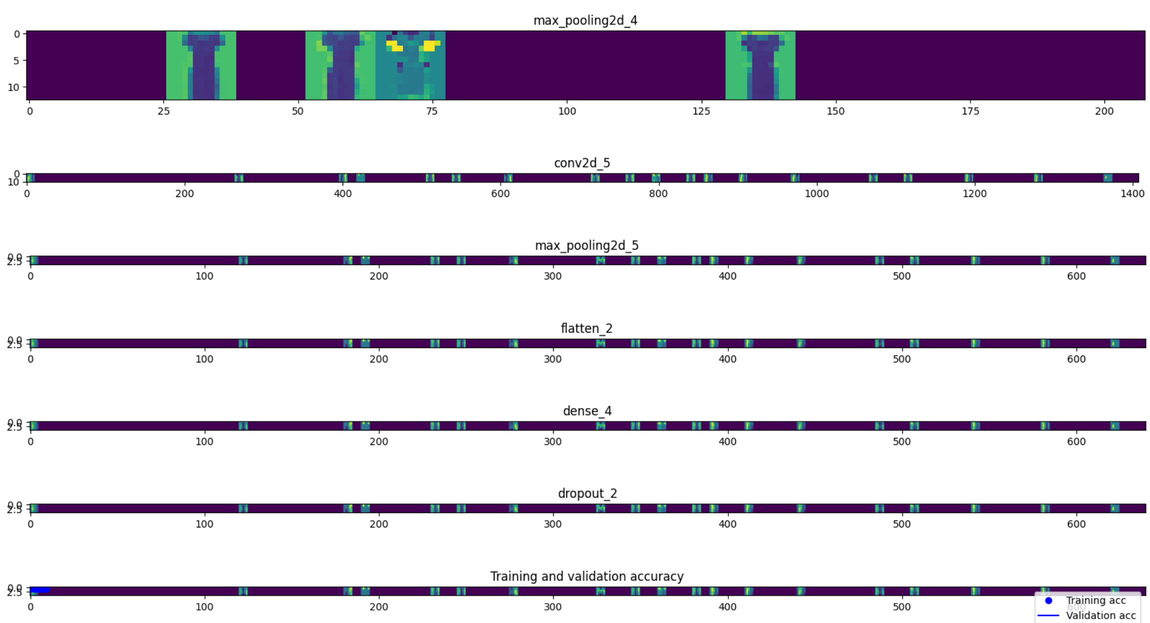
Prediction is -> [-5.1830626 -6.8902445 -8.186485 -2.54921 -11.686236 3.018158

-11.358143 4.5767307 6.5097156 -3.2124116]

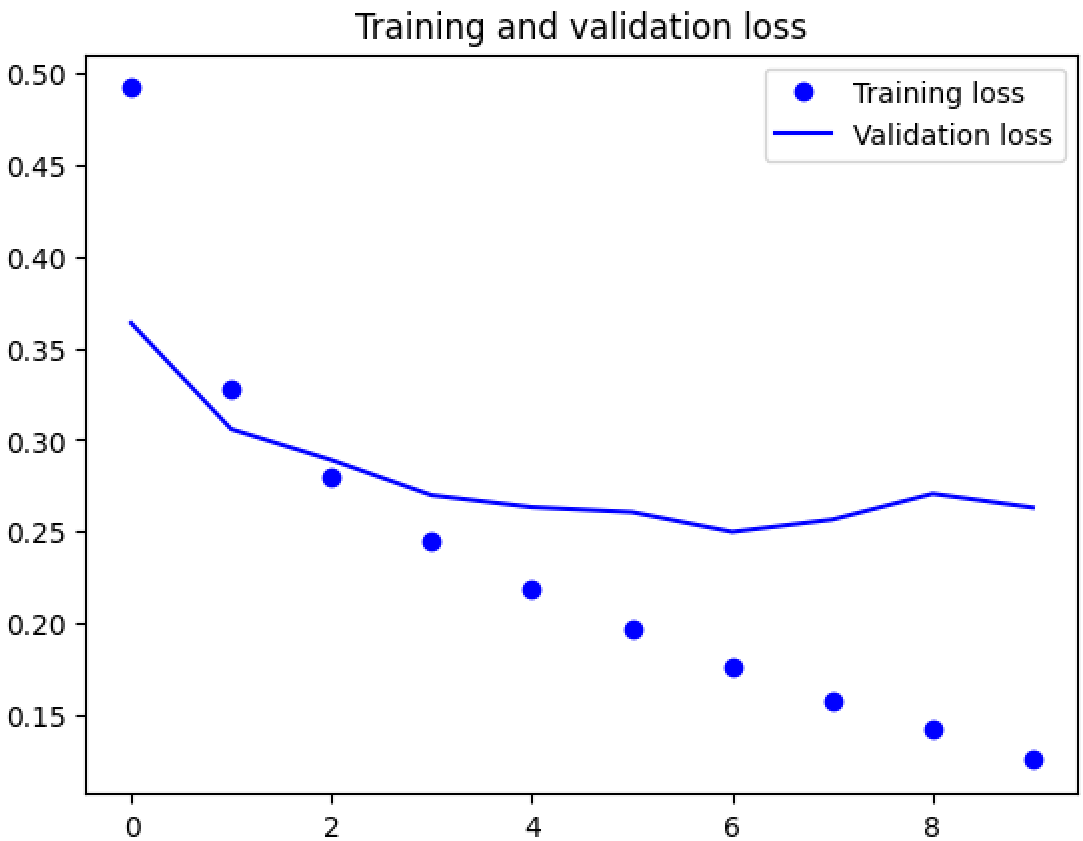
Actual value is -> 7

The highest value for label is 8





max_pooling2d_4
conv2d_5
max_pooling2d_5
flatten_2
dense_4
dropout_2
Training and validation accuracy
Training acc
Validation acc


Training and validation loss
Training loss
Validation loss
0.50
0.45
0.40
0.35
0.30
0.25
0.20
0.15
0
2
4
6
8