Capstone Project

Image classifier for the SVHN dataset

Instructions

In this notebook, you will create a neural network that classifies real-world images digits. You will use concepts from throughout this course in building, training, testing, validating and saving your Tensorflow classifier model.

This project is peer-assessed. Within this notebook you will find instructions in each section for how to complete the project. Pay close attention to the instructions as the peer review will be carried out according to a grading rubric that checks key parts of the project instructions. Feel free to add extra cells into the notebook as required.

How to submit

When you have completed the Capstone project notebook, you will submit a pdf of the notebook for peer review. First ensure that the notebook has been fully executed from beginning to end, and all of the cell outputs are visible. This is important, as the grading rubric depends on the reviewer being able to view the outputs of your notebook. Save the notebook as a pdf (File -> Download as -> PDF via LaTeX). You should then submit this pdf for review.

Let's get started!

We'll start by running some imports, and loading the dataset. For this project you are free to make further imports throughout the notebook as you wish.

In [ ]:
```
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Flatten, Softmax, Conv2D, MaxPooling2D, Drc
from tensorflow.keras.callbacks import ModelCheckpoint
from scipy.io import loadmat
from matplotlib import pyplot as plt
import numpy as np
import pandas as pd
```

For the capstone project, you will use the [SVHN dataset](). This is an image dataset of over 600,000 digit images in all, and is a harder dataset than MNIST as the numbers appear in the context of natural scene images. SVHN is obtained from house numbers in Google Street View images.

- Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu and A. Y. Ng. "Reading Digits in Natural Images with Unsupervised Feature Learning". NIPS Workshop on Deep Learning and Unsupervised Feature Learning, 2011.

Your goal is to develop an end-to-end workflow for building, training, validating, evaluating and saving a neural network that classifies a real-world image into one of ten classes.

In [ ]:
```
# Run this cell to load the dataset

train = loadmat('data/train_32x32.mat')
test = loadmat('data/test_32x32.mat')
```

Both `train` and `test` are dictionaries with keys `X` and `y` for the input images and labels respectively.

1. Inspect and preprocess the dataset

- Extract the training and testing images and labels separately from the train and test dictionaries loaded for you.
- Select a random sample of images and corresponding labels from the dataset (at least 10), and display them in a figure.
- Convert the training and test images to grayscale by taking the average across all colour channels for each pixel. *Hint: retain the channel dimension, which will now have size 1.*
- Select a random sample of the grayscale images and corresponding labels from the dataset (at least 10), and display them in a figure.

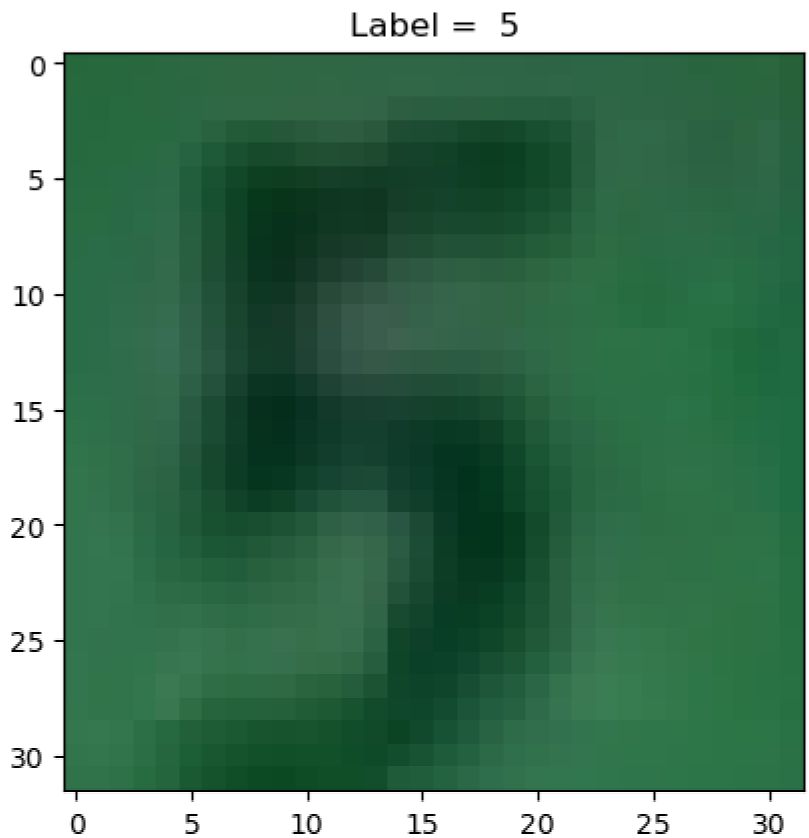
In [ ]:
```
x_train, y_train = train['X'], train['y']
x_test, y_test = test['X'], test['y']
```

In [ ]:
```
x_test.shape, x_train.shape
```

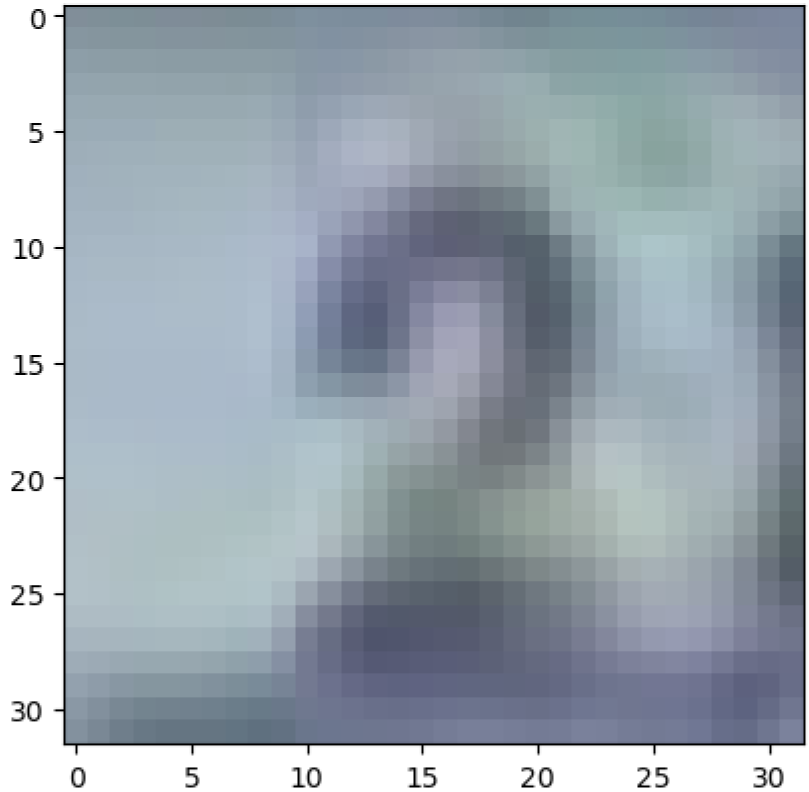
Out[ ]:
```
((32, 32, 3, 26032), (32, 32, 3, 73257))
```

In [ ]:
```
x_train = np.transpose(x_train, (3, 0, 1, 2))
x_test = np.transpose(x_test, (3, 0, 1, 2))
```

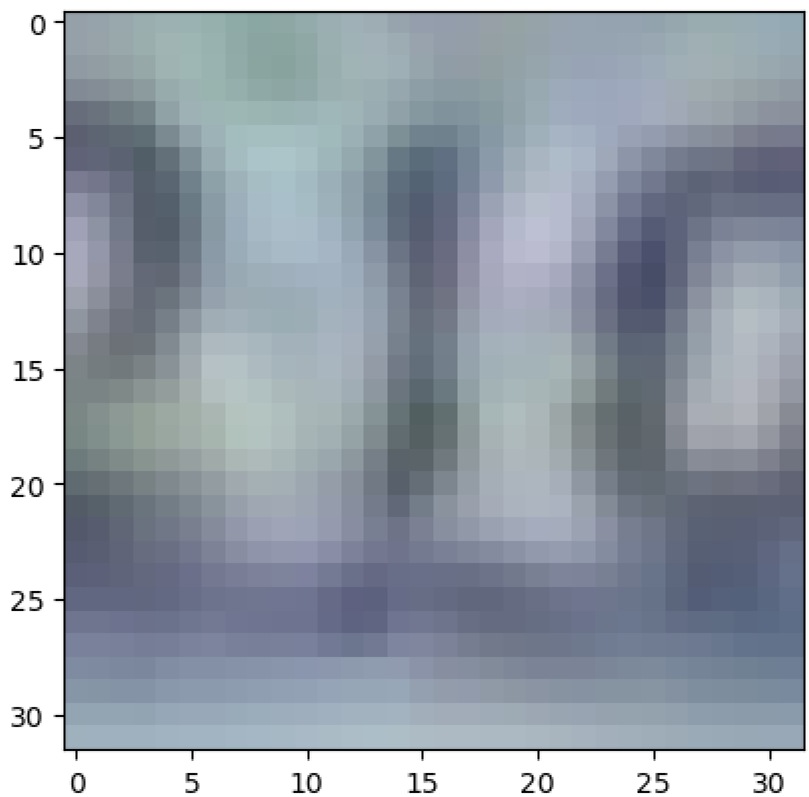
In [ ]:
```
for i in range(0,11):
    fig = plt.figure()
    plt.title('Label =  %i' %(int(y_test[i])))
    plt.imshow(x_test[i])
    plt.show()
```

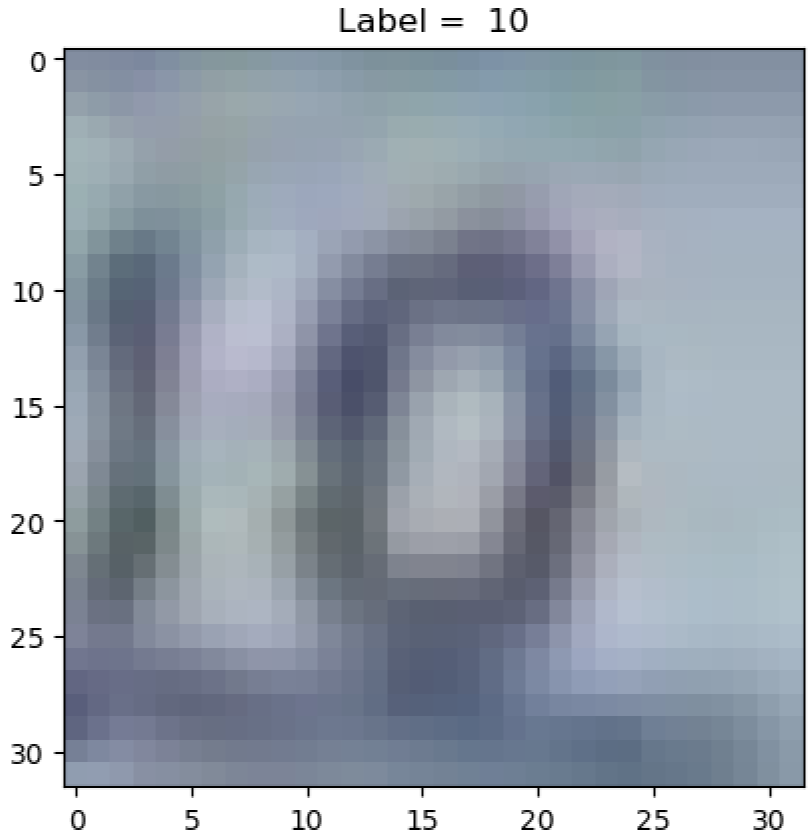


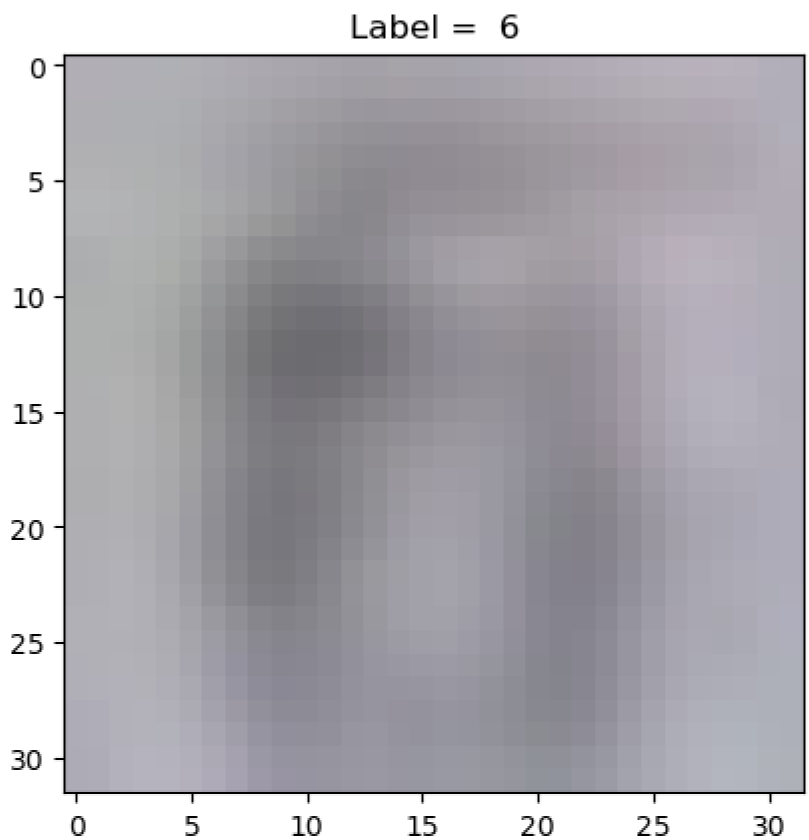
Label = 2

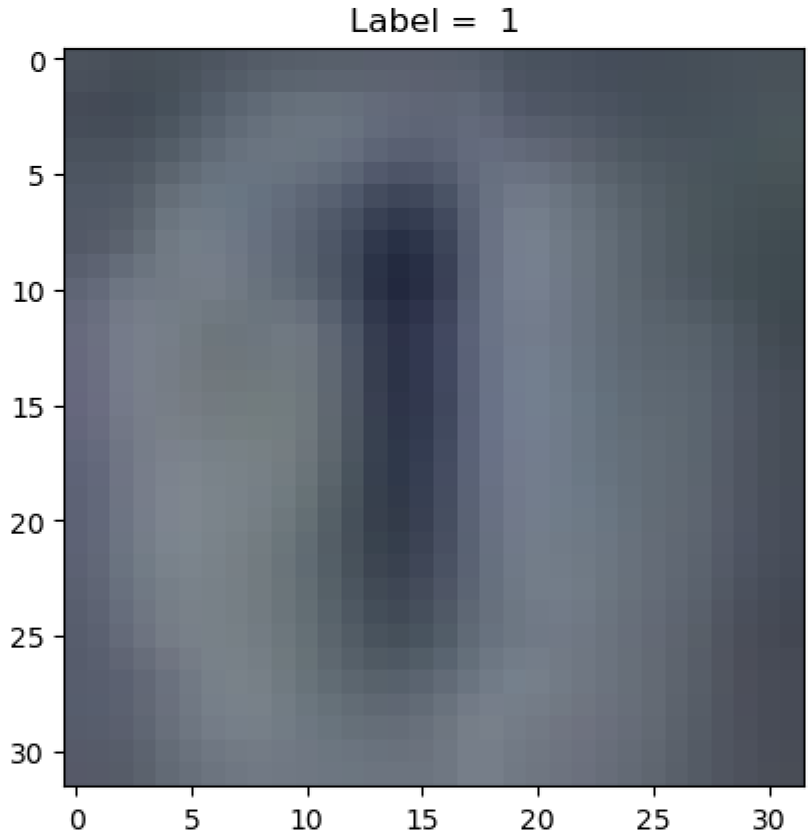


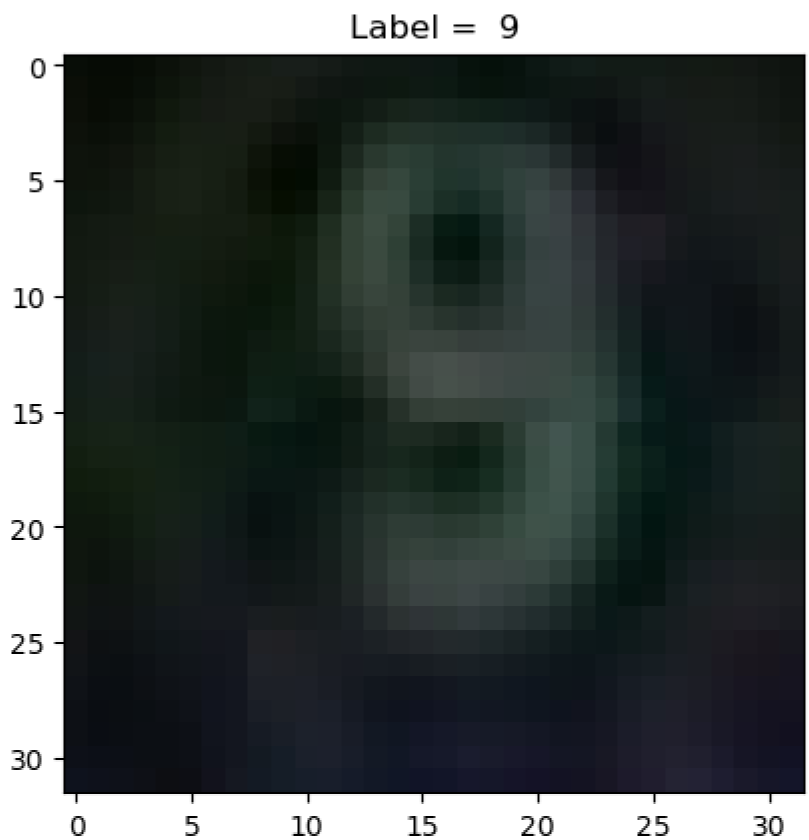
Label = 1

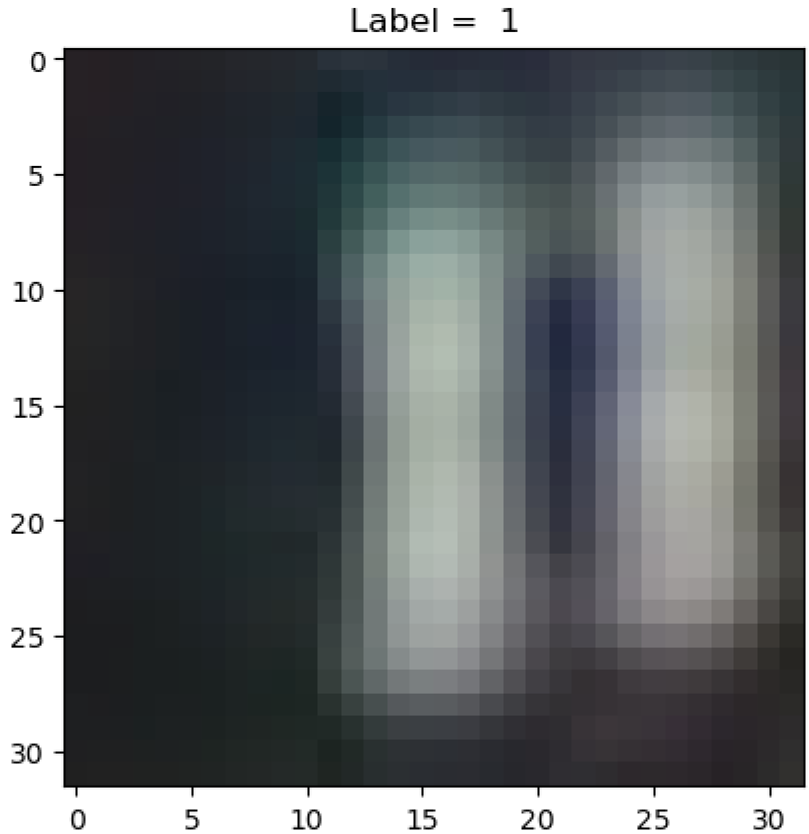


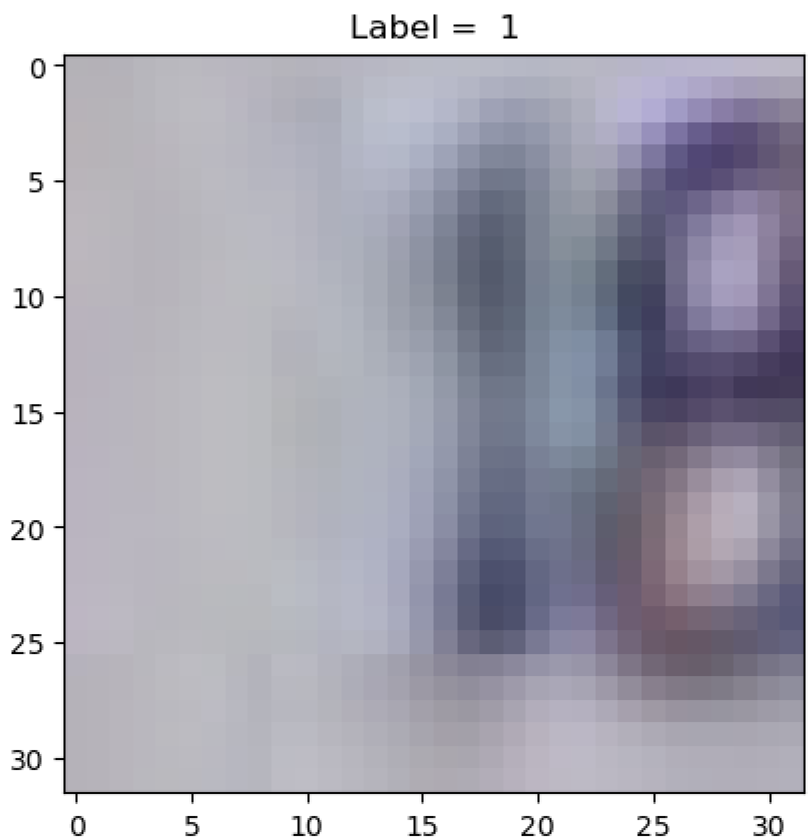

Label = 10
0
5
10
15
20
25
30
0
5
10
15
20
25
30

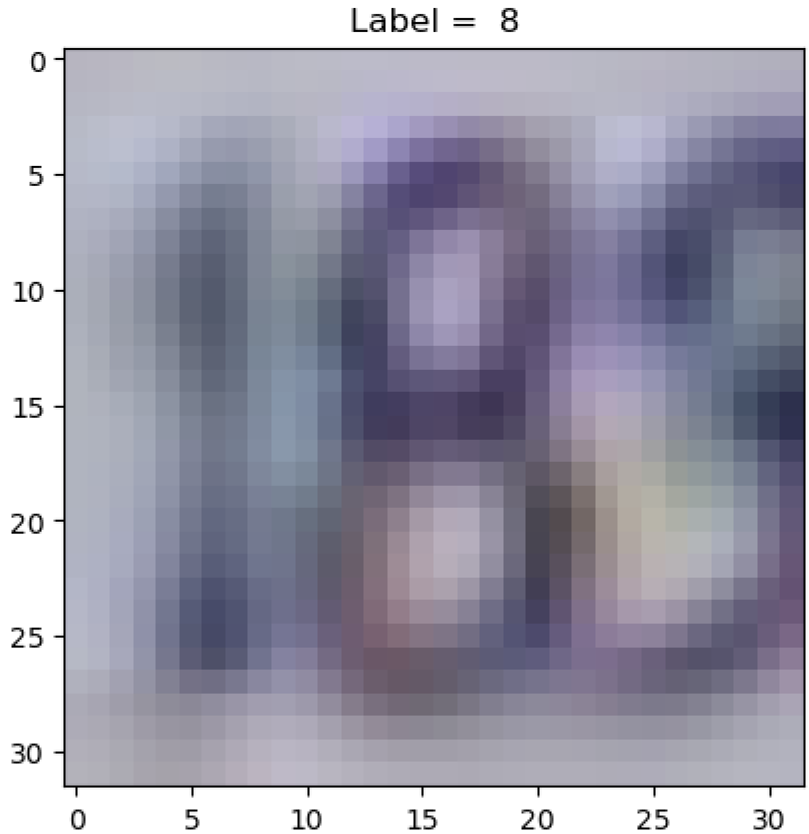

Label = 6
0
5
10
15
20
25
30
0
5
10
15
20
25
30

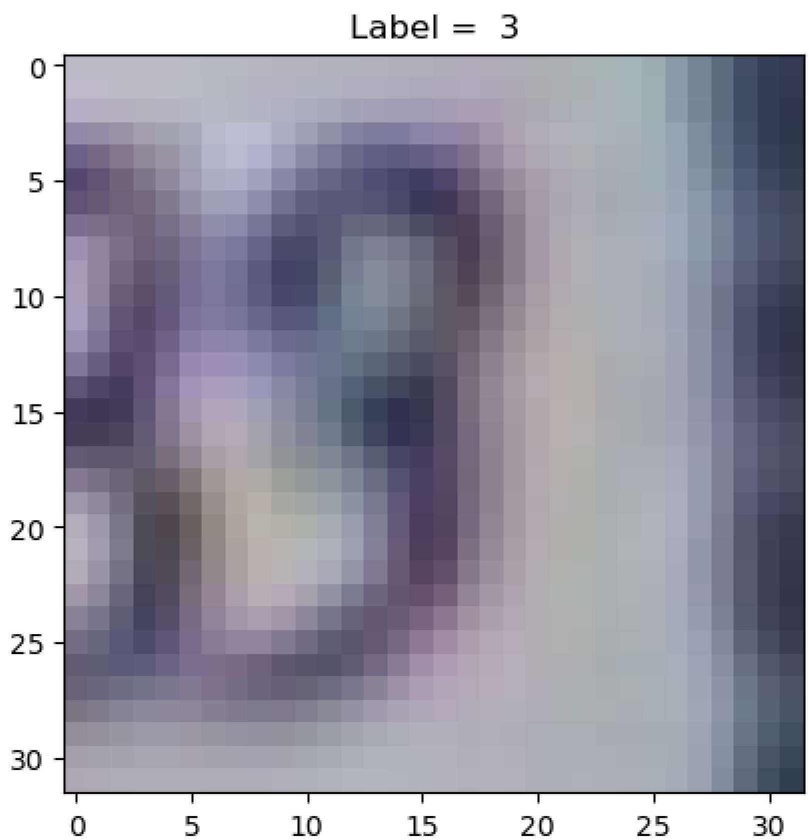

Label = 1
0
5
10
15
20
25
30
0
5
10
15
20
25
30


Label = 9
0
5
10
15
20
25
30
0
5
10
15
20
25
30


Label = 1
0
5
10
15
20
25
30
0
5
10
15
20
25
30


Label = 1
0
5
10
15
20
25
30
0
5
10
15
20
25
30

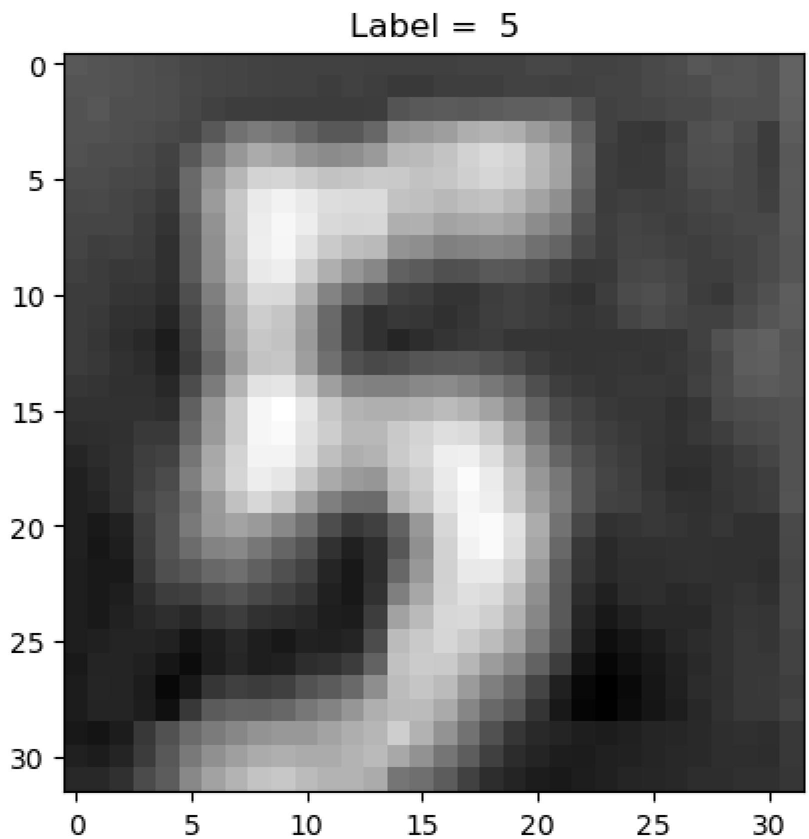


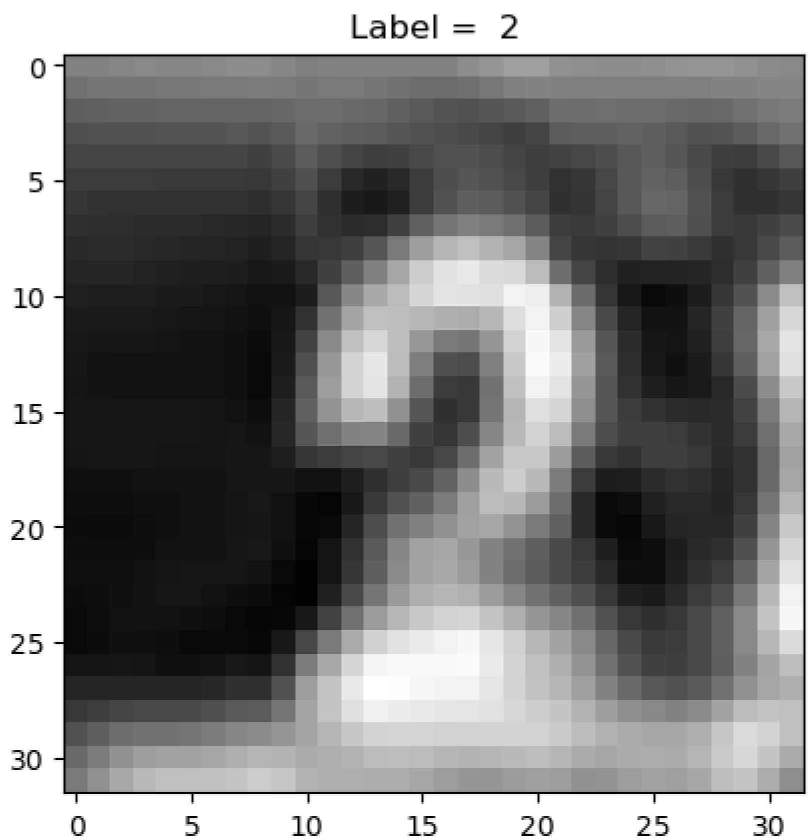


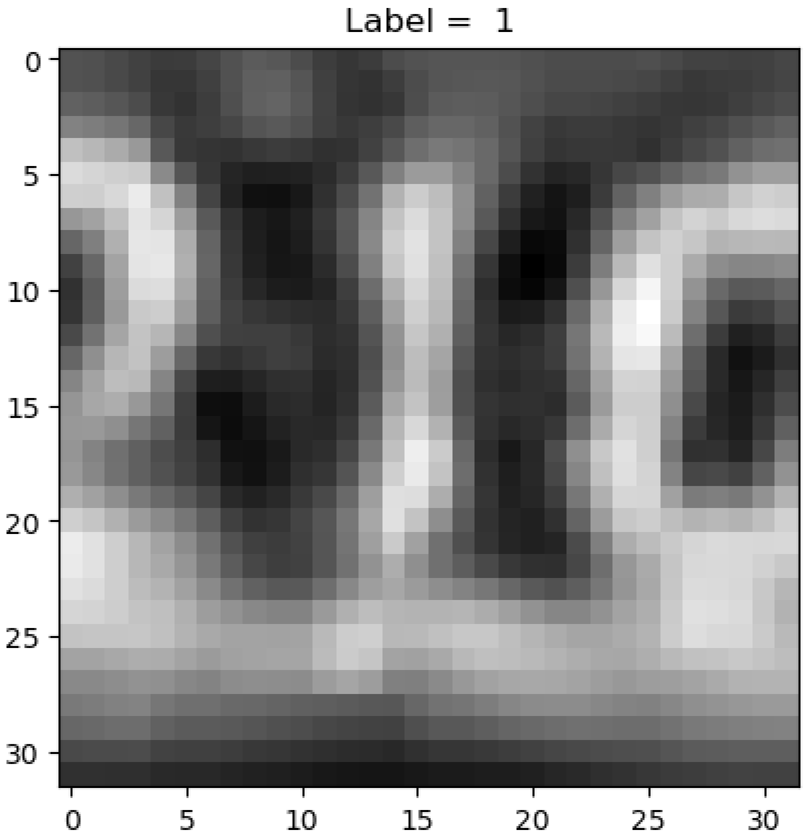
In [ ]:
```
x_test = np.sum(x_test, axis=3) / 3
x_train = np.sum(x_train, axis=3) / 3
```

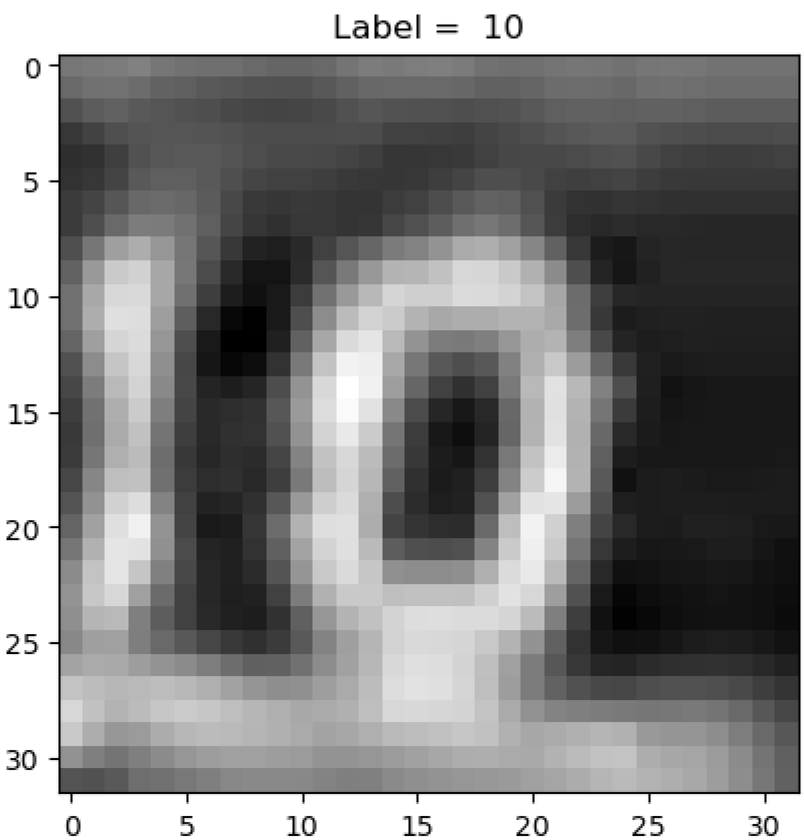
In [ ]:
```
for i in range(0,11):
    fig = plt.figure()
```

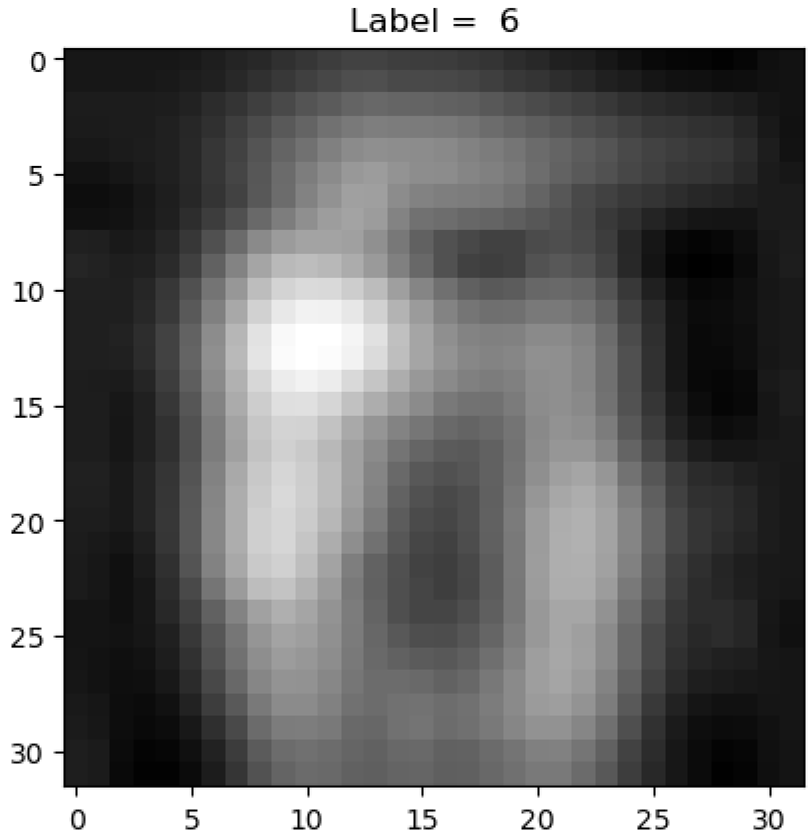
```
    plt.title('Label =  %i' %(int(y_test[i])))
    plt.imshow(x_test[i], cmap='gray_r')
    plt.show()
```

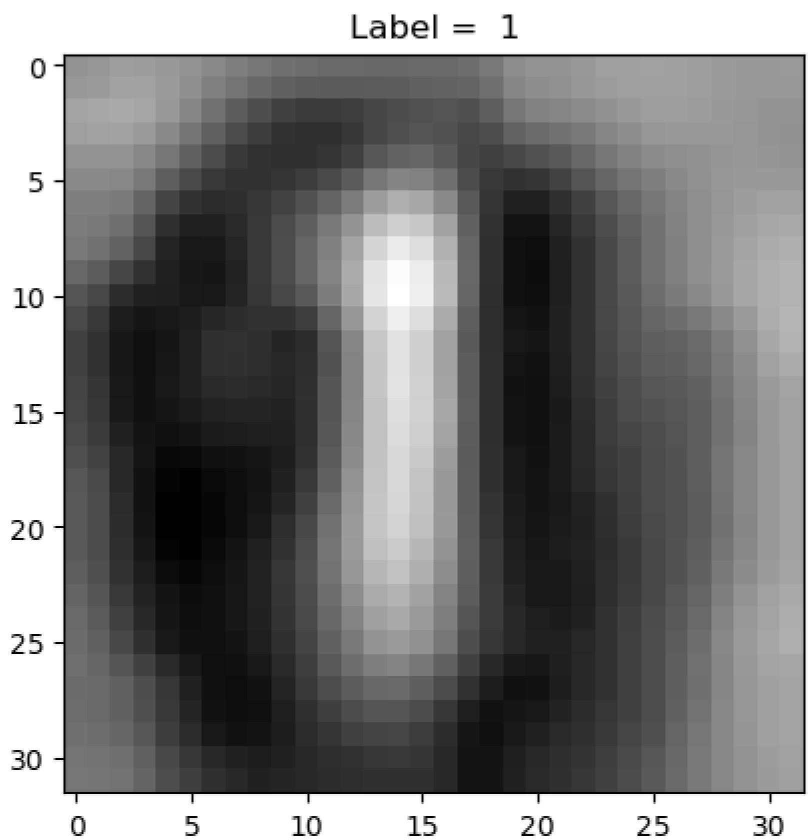


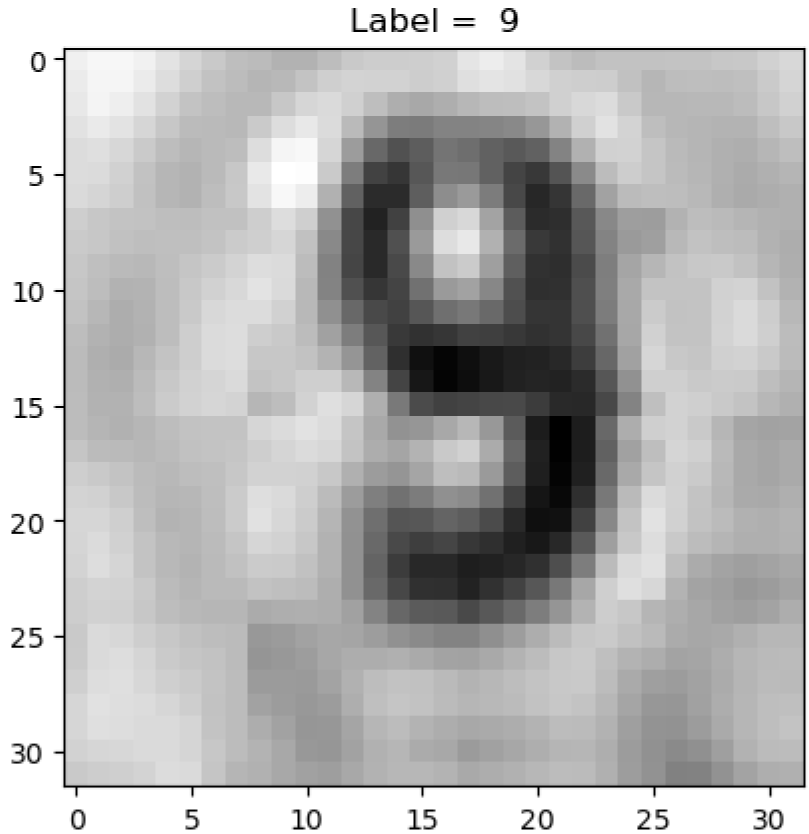


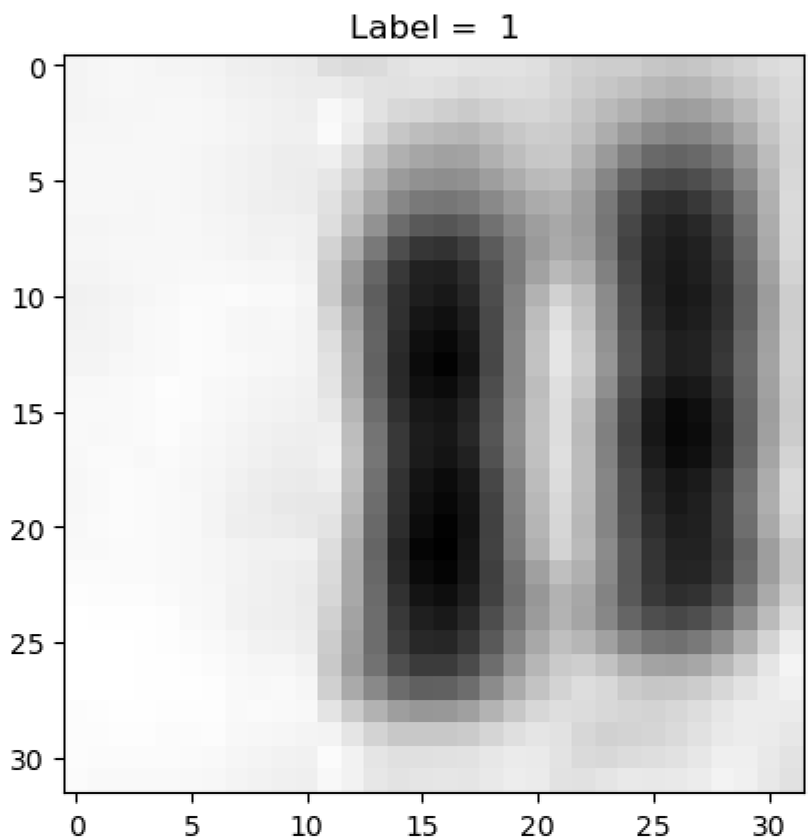

Label = 1

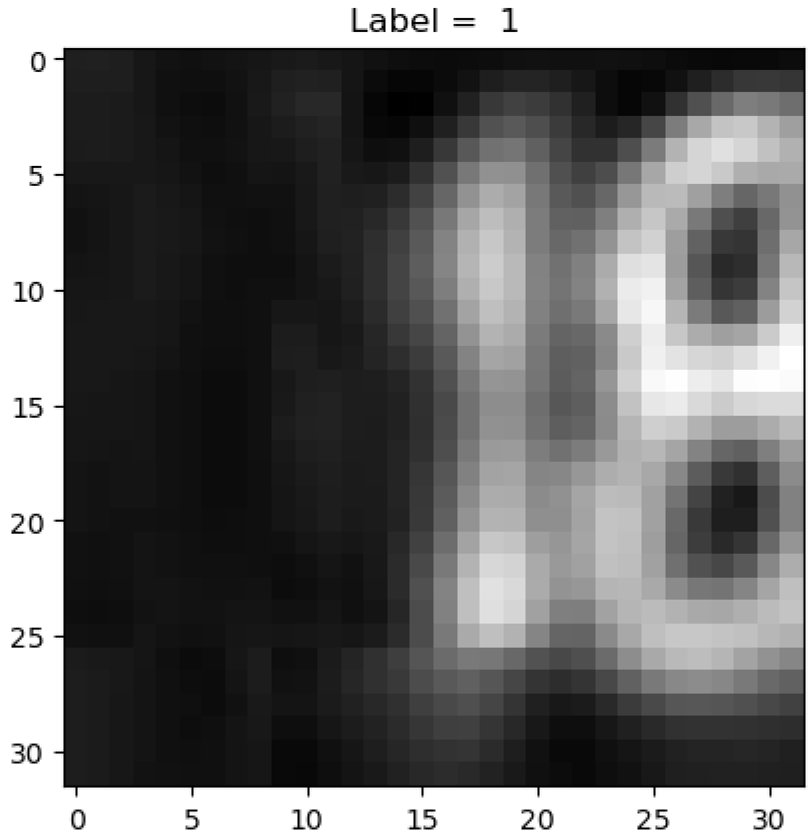

Label = 10

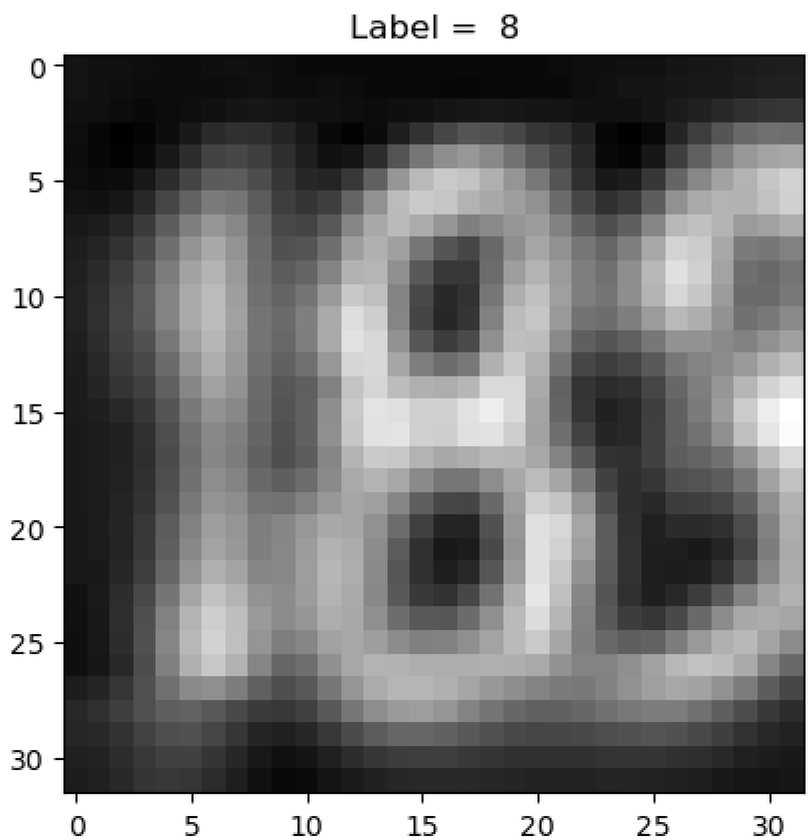

Label = 6
0
5
10
15
20
25
30
0
5
10
15
20
25
30

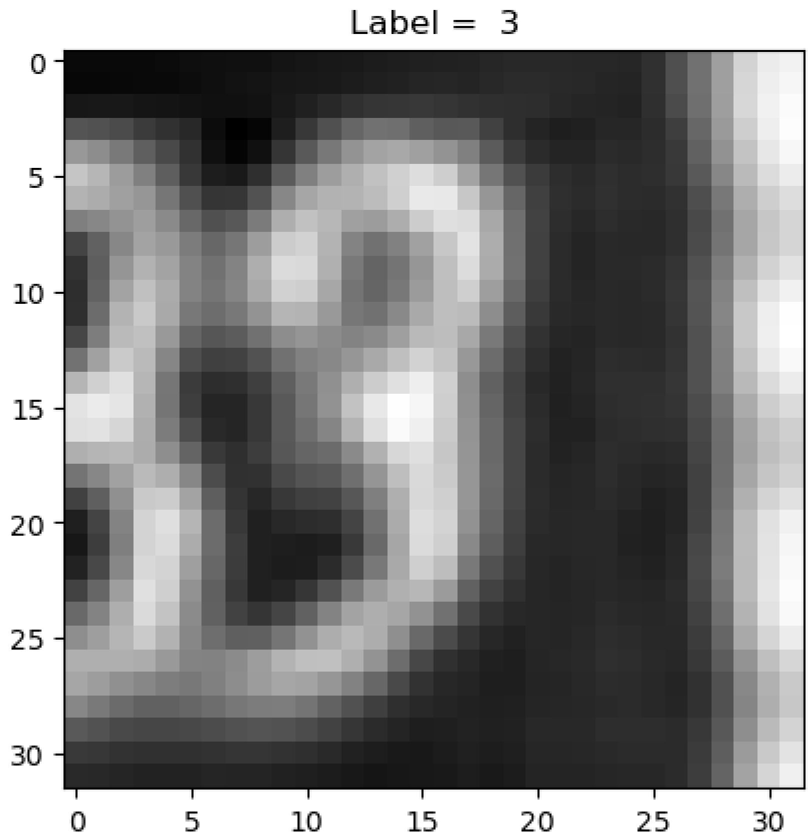

Label = 1
0
5
10
15
20
25
30
0
5
10
15
20
25
30


Label = 9
0
5
10
15
20
25
30
0
5
10
15
20
25
30


Label = 1
0
5
10
15
20
25
30
0
5
10
15
20
25
30


Label = 1
0
5
10
15
20
25
30
0
5
10
15
20
25
30


Label = 8
0
5
10
15
20
25
30
0
5
10
15
20
25
30



2. MLP neural network classifier

- Build an MLP classifier model using the Sequential API. Your model should use only Flatten and Dense layers, with the final layer having a 10-way softmax output.
- You should design and build the model yourself. Feel free to experiment with different MLP architectures. *Hint: to achieve a reasonable accuracy you won't need to use more than 4 or 5 layers.*
- Print out the model summary (using the summary() method)
- Compile and train the model (we recommend a maximum of 30 epochs), making use of both training and validation sets during the training run.
- Your model should track at least one appropriate metric, and use at least two callbacks during training, one of which should be a ModelCheckpoint callback.
- As a guide, you should aim to achieve a final categorical cross entropy training loss of less than 1.0 (the validation loss might be higher).
- Plot the learning curves for loss vs epoch and accuracy vs epoch for both training and validation sets.
- Compute and display the loss and accuracy of the trained model on the test set.

In [ ]:
```
# Developing the Sequential model for MLP classifier
model_mlp = Sequential([
    Flatten(input_shape=(32, 32, 1)),
    Dense(128, activation='relu'),
    Dense(64, activation='relu'),
    Dense(64, activation='relu'),
    Dense(32, activation='relu'),
```

```
    Dense(11, activation='softmax')
])
```

In [ ]:
```
model_mlp.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 flatten (Flatten)           (None, 1024)              0         
                                                                 
 dense (Dense)               (None, 128)               131200    
                                                                 
 dense_1 (Dense)             (None, 64)                8256      
                                                                 
 dense_2 (Dense)             (None, 64)                4160      
                                                                 
 dense_3 (Dense)             (None, 32)                2080      
                                                                 
 dense_4 (Dense)             (None, 11)                363       
                                                                 
=================================================================
Total params: 146,059
Trainable params: 146,059
Non-trainable params: 0
_________________________________________________________________
```

In [ ]:
```
# Compiling the MLP model
model_mlp.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['
```

In [ ]:
```
# Defining the Callbacks
checkpoint_path_mlp = 'model_checkpoints/checkpoint'
checkpoint_mlp = ModelCheckpoint(checkpoint_path_mlp, frequency='epoch', save_weights_
                             verbose=1)
early_stopping = tf.keras.callbacks.EarlyStopping(monitor='accuracy',patience=2)
```

In [ ]:
```
# Reshaping the datasets

x_train = x_train[...,np.newaxis]
x_test = x_test[...,np.newaxis]
```

In [ ]:
```
# Fitting the MLP model

history_mlp = model_mlp.fit(x_train, y_train, epochs=30, batch_size=260, validation_sp
```

```
Epoch 1/30
254/254 [==============================] - ETA: 0s - loss: 4.9138 - accuracy: 0.1297
Epoch 1: val_loss improved from inf to 2.36020, saving model to model_checkpoints\che
ckpoint
254/254 [==============================] - 5s 14ms/step - loss: 4.9138 - accuracy: 0.
1297 - val_loss: 2.3602 - val_accuracy: 0.1863
Epoch 2/30
249/254 [============================>.] - ETA: 0s - loss: 2.3279 - accuracy: 0.1797
Epoch 2: val_loss improved from 2.36020 to 2.31391, saving model to model_checkpoints
\checkpoint
254/254 [==============================] - 2s 10ms/step - loss: 2.3275 - accuracy: 0.
1800 - val_loss: 2.3139 - val_accuracy: 0.1766
Epoch 3/30
254/254 [==============================] - ETA: 0s - loss: 2.2710 - accuracy: 0.1919
Epoch 3: val_loss improved from 2.31391 to 2.24837, saving model to model_checkpoints
\checkpoint
254/254 [==============================] - 2s 9ms/step - loss: 2.2710 - accuracy: 0.1
919 - val_loss: 2.2484 - val_accuracy: 0.1946
Epoch 4/30
251/254 [============================>.] - ETA: 0s - loss: 2.2053 - accuracy: 0.2108
Epoch 4: val_loss improved from 2.24837 to 2.19060, saving model to model_checkpoints
\checkpoint
254/254 [==============================] - 2s 10ms/step - loss: 2.2049 - accuracy: 0.
2109 - val_loss: 2.1906 - val_accuracy: 0.2140
Epoch 5/30
250/254 [============================>.] - ETA: 0s - loss: 2.1489 - accuracy: 0.2358
Epoch 5: val_loss improved from 2.19060 to 2.18165, saving model to model_checkpoints
\checkpoint
254/254 [==============================] - 2s 10ms/step - loss: 2.1488 - accuracy: 0.
2356 - val_loss: 2.1816 - val_accuracy: 0.1933
Epoch 6/30
250/254 [============================>.] - ETA: 0s - loss: 2.1002 - accuracy: 0.2628
Epoch 6: val_loss improved from 2.18165 to 2.05045, saving model to model_checkpoints
\checkpoint
254/254 [==============================] - 2s 8ms/step - loss: 2.0999 - accuracy: 0.2
628 - val_loss: 2.0505 - val_accuracy: 0.2891
Epoch 7/30
250/254 [============================>.] - ETA: 0s - loss: 2.0311 - accuracy: 0.2937
Epoch 7: val_loss improved from 2.05045 to 2.00536, saving model to model_checkpoints
\checkpoint
254/254 [==============================] - 2s 8ms/step - loss: 2.0312 - accuracy: 0.2
938 - val_loss: 2.0054 - val_accuracy: 0.3008
Epoch 8/30
251/254 [============================>.] - ETA: 0s - loss: 2.0000 - accuracy: 0.2958
Epoch 8: val_loss improved from 2.00536 to 1.95475, saving model to model_checkpoints
\checkpoint
254/254 [==============================] - 2s 8ms/step - loss: 1.9996 - accuracy: 0.2
958 - val_loss: 1.9548 - val_accuracy: 0.2991
Epoch 9/30
251/254 [============================>.] - ETA: 0s - loss: 1.8828 - accuracy: 0.3339
Epoch 9: val_loss improved from 1.95475 to 1.88399, saving model to model_checkpoints
\checkpoint
254/254 [==============================] - 2s 9ms/step - loss: 1.8820 - accuracy: 0.3
342 - val_loss: 1.8840 - val_accuracy: 0.3445
Epoch 10/30
250/254 [============================>.] - ETA: 0s - loss: 1.8032 - accuracy: 0.3819
Epoch 10: val_loss improved from 1.88399 to 1.77925, saving model to model_checkpoint
s\checkpoint
254/254 [==============================] - 2s 8ms/step - loss: 1.8019 - accuracy: 0.3
822 - val_loss: 1.7793 - val_accuracy: 0.3990
```

```
Epoch 11/30
254/254 [==============================] - ETA: 0s - loss: 1.7268 - accuracy: 0.4207
Epoch 11: val_loss improved from 1.77925 to 1.67799, saving model to model_checkpoint
s\checkpoint
254/254 [==============================] - 2s 9ms/step - loss: 1.7268 - accuracy: 0.4
207 - val_loss: 1.6780 - val_accuracy: 0.4323
Epoch 12/30
254/254 [==============================] - ETA: 0s - loss: 1.5912 - accuracy: 0.4651
Epoch 12: val_loss improved from 1.67799 to 1.51520, saving model to model_checkpoint
s\checkpoint
254/254 [==============================] - 2s 8ms/step - loss: 1.5912 - accuracy: 0.4
651 - val_loss: 1.5152 - val_accuracy: 0.4803
Epoch 13/30
249/254 [============================>.] - ETA: 0s - loss: 1.4290 - accuracy: 0.5255
Epoch 13: val_loss improved from 1.51520 to 1.35996, saving model to model_checkpoint
s\checkpoint
254/254 [==============================] - 2s 9ms/step - loss: 1.4285 - accuracy: 0.5
260 - val_loss: 1.3600 - val_accuracy: 0.5468
Epoch 14/30
252/254 [============================>.] - ETA: 0s - loss: 1.3181 - accuracy: 0.5784
Epoch 14: val_loss improved from 1.35996 to 1.24938, saving model to model_checkpoint
s\checkpoint
254/254 [==============================] - 4s 17ms/step - loss: 1.3175 - accuracy: 0.
5788 - val_loss: 1.2494 - val_accuracy: 0.6088
Epoch 15/30
253/254 [============================>.] - ETA: 0s - loss: 1.2468 - accuracy: 0.6064
Epoch 15: val_loss did not improve from 1.24938
254/254 [==============================] - 3s 13ms/step - loss: 1.2470 - accuracy: 0.
6063 - val_loss: 1.2679 - val_accuracy: 0.5979
Epoch 16/30
251/254 [============================>.] - ETA: 0s - loss: 1.1928 - accuracy: 0.6266
Epoch 16: val_loss improved from 1.24938 to 1.16893, saving model to model_checkpoint
s\checkpoint
254/254 [==============================] - 4s 15ms/step - loss: 1.1923 - accuracy: 0.
6268 - val_loss: 1.1689 - val_accuracy: 0.6226
Epoch 17/30
252/254 [============================>.] - ETA: 0s - loss: 1.1414 - accuracy: 0.6461
Epoch 17: val_loss improved from 1.16893 to 1.15735, saving model to model_checkpoint
s\checkpoint
254/254 [==============================] - 3s 12ms/step - loss: 1.1410 - accuracy: 0.
6461 - val_loss: 1.1573 - val_accuracy: 0.6373
Epoch 18/30
251/254 [============================>.] - ETA: 0s - loss: 1.1067 - accuracy: 0.6580
Epoch 18: val_loss improved from 1.15735 to 1.04085, saving model to model_checkpoint
s\checkpoint
254/254 [==============================] - 3s 13ms/step - loss: 1.1060 - accuracy: 0.
6581 - val_loss: 1.0408 - val_accuracy: 0.6796
Epoch 19/30
251/254 [============================>.] - ETA: 0s - loss: 1.0641 - accuracy: 0.6733
Epoch 19: val_loss did not improve from 1.04085
254/254 [==============================] - 3s 12ms/step - loss: 1.0640 - accuracy: 0.
6732 - val_loss: 1.1133 - val_accuracy: 0.6493
Epoch 20/30
249/254 [============================>.] - ETA: 0s - loss: 1.0648 - accuracy: 0.6722
Epoch 20: val_loss did not improve from 1.04085
254/254 [==============================] - 3s 11ms/step - loss: 1.0646 - accuracy: 0.
6725 - val_loss: 1.0526 - val_accuracy: 0.6749
Epoch 21/30
251/254 [============================>.] - ETA: 0s - loss: 1.0211 - accuracy: 0.6878
Epoch 21: val_loss did not improve from 1.04085
```

```
254/254 [==============================] - 3s 11ms/step - loss: 1.0208 - accuracy: 0.
6877 - val_loss: 1.0864 - val_accuracy: 0.6626
Epoch 22/30
252/254 [============================>.] - ETA: 0s - loss: 1.0206 - accuracy: 0.6891
Epoch 22: val_loss improved from 1.04085 to 0.98228, saving model to model_checkpoint
s\checkpoint
254/254 [==============================] - 3s 11ms/step - loss: 1.0210 - accuracy: 0.
6890 - val_loss: 0.9823 - val_accuracy: 0.7027
Epoch 23/30
252/254 [============================>.] - ETA: 0s - loss: 0.9803 - accuracy: 0.7021
Epoch 23: val_loss improved from 0.98228 to 0.95524, saving model to model_checkpoint
s\checkpoint
254/254 [==============================] - 3s 11ms/step - loss: 0.9807 - accuracy: 0.
7021 - val_loss: 0.9552 - val_accuracy: 0.7139
Epoch 24/30
251/254 [============================>.] - ETA: 0s - loss: 0.9809 - accuracy: 0.7016
Epoch 24: val_loss did not improve from 0.95524
254/254 [==============================] - 3s 12ms/step - loss: 0.9802 - accuracy: 0.
7020 - val_loss: 0.9609 - val_accuracy: 0.7125
Epoch 25/30
252/254 [============================>.] - ETA: 0s - loss: 0.9633 - accuracy: 0.7075
Epoch 25: val_loss did not improve from 0.95524
254/254 [==============================] - 3s 11ms/step - loss: 0.9631 - accuracy: 0.
7075 - val_loss: 0.9781 - val_accuracy: 0.7030
Epoch 26/30
253/254 [============================>.] - ETA: 0s - loss: 0.9416 - accuracy: 0.7116
Epoch 26: val_loss did not improve from 0.95524
254/254 [==============================] - 3s 11ms/step - loss: 0.9412 - accuracy: 0.
7118 - val_loss: 0.9726 - val_accuracy: 0.7058
Epoch 27/30
253/254 [============================>.] - ETA: 0s - loss: 0.9306 - accuracy: 0.7169
Epoch 27: val_loss did not improve from 0.95524
254/254 [==============================] - 3s 11ms/step - loss: 0.9310 - accuracy: 0.
7167 - val_loss: 1.1395 - val_accuracy: 0.6403
Epoch 28/30
253/254 [============================>.] - ETA: 0s - loss: 0.9344 - accuracy: 0.7157
Epoch 28: val_loss improved from 0.95524 to 0.89335, saving model to model_checkpoint
s\checkpoint
254/254 [==============================] - 3s 11ms/step - loss: 0.9345 - accuracy: 0.
7157 - val_loss: 0.8933 - val_accuracy: 0.7305
Epoch 29/30
253/254 [============================>.] - ETA: 0s - loss: 0.9054 - accuracy: 0.7243
Epoch 29: val_loss did not improve from 0.89335
254/254 [==============================] - 3s 14ms/step - loss: 0.9051 - accuracy: 0.
7244 - val_loss: 0.9547 - val_accuracy: 0.7065
Epoch 30/30
251/254 [============================>.] - ETA: 0s - loss: 0.8955 - accuracy: 0.7292
Epoch 30: val_loss did not improve from 0.89335
254/254 [==============================] - 4s 15ms/step - loss: 0.8958 - accuracy: 0.
7291 - val_loss: 0.9911 - val_accuracy: 0.6942
```

In [ ]:

```
# Converting the history to pandas dataframe

df_mlp = pd.DataFrame(history_mlp.history)
df_mlp.head()
```

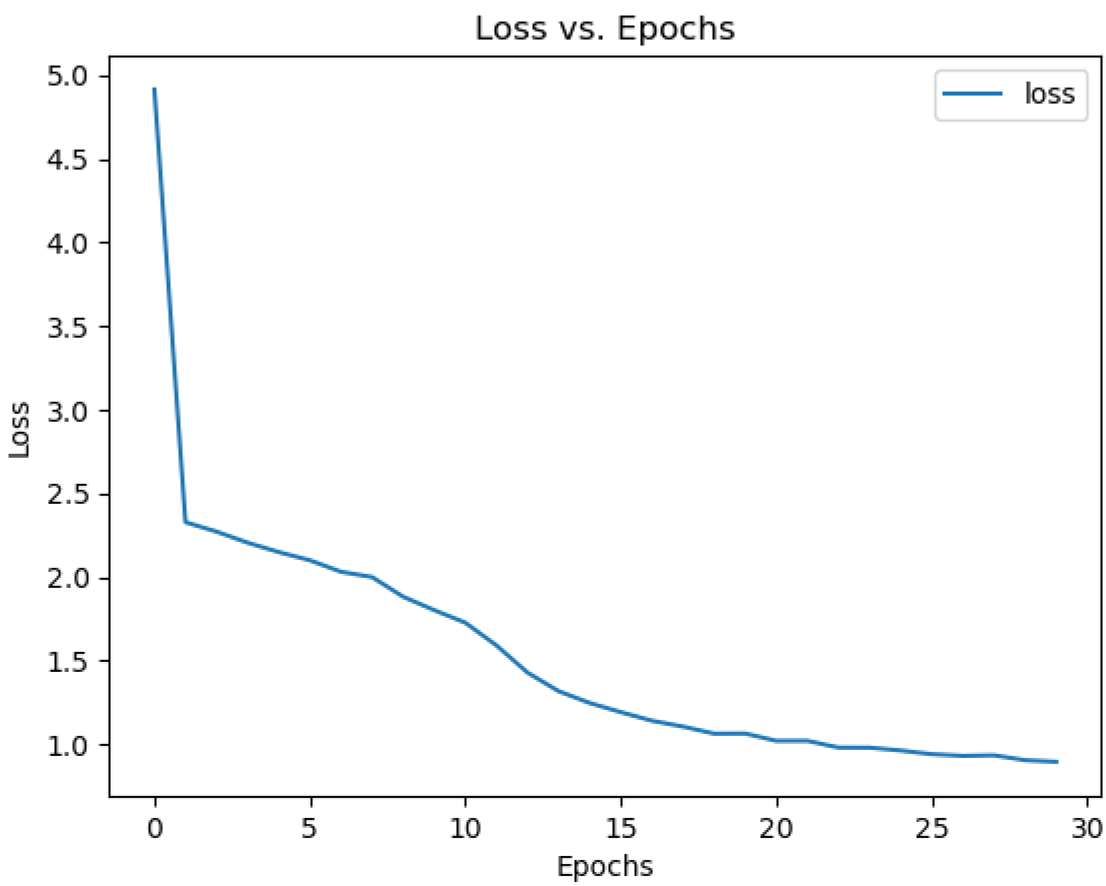
Out[ ]:

	loss	accuracy	val_loss	val_accuracy
0	4.913829	0.129666	2.360201	0.186323
1	2.327513	0.179976	2.313905	0.176631
2	2.270968	0.191898	2.248368	0.194649
3	2.204926	0.210857	2.190604	0.214032
4	2.148843	0.235595	2.181646	0.193284

In [ ]:

```
# Make a plot for the loss

loss_plot = df_mlp.plot(y='loss', title='Loss vs. Epochs', legend='False')
loss_plot.set(xlabel='Epochs', ylabel='Loss')
```

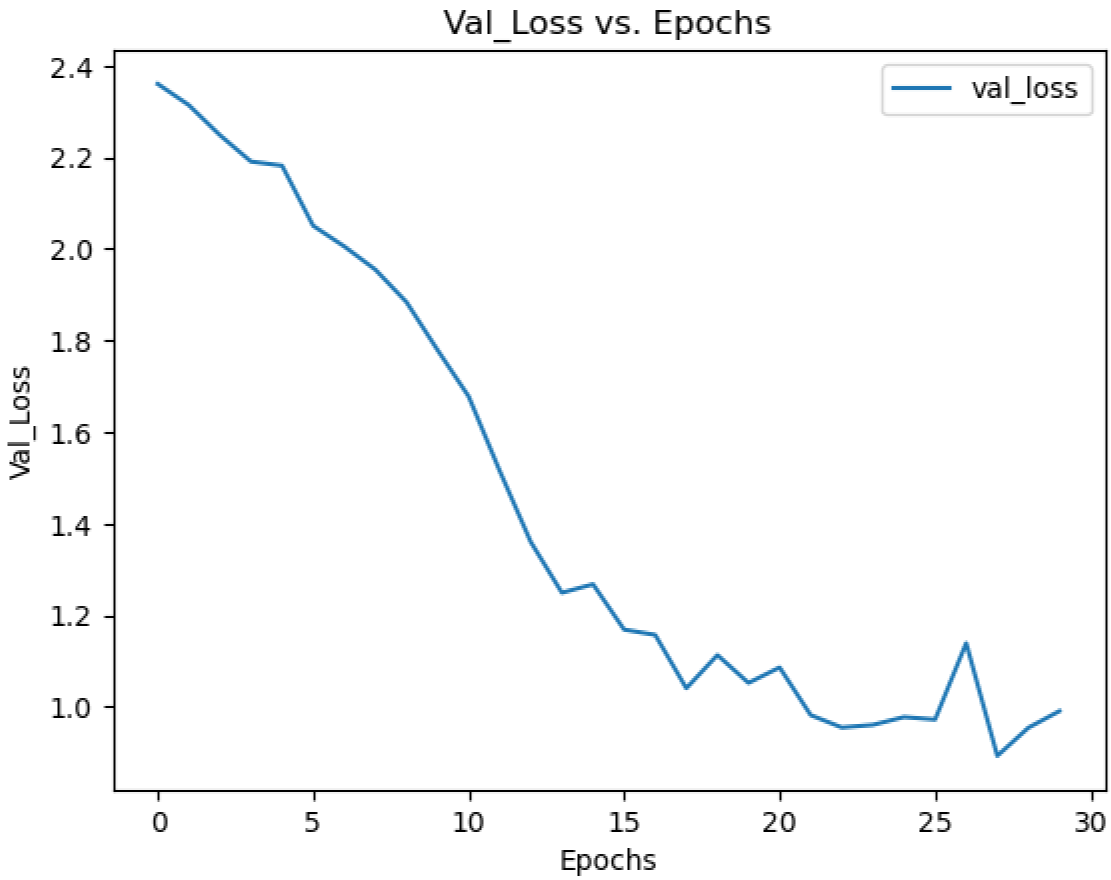
Out[ ]: [Text(0.5, 0, 'Epochs'), Text(0, 0.5, 'Loss')]



In [ ]:

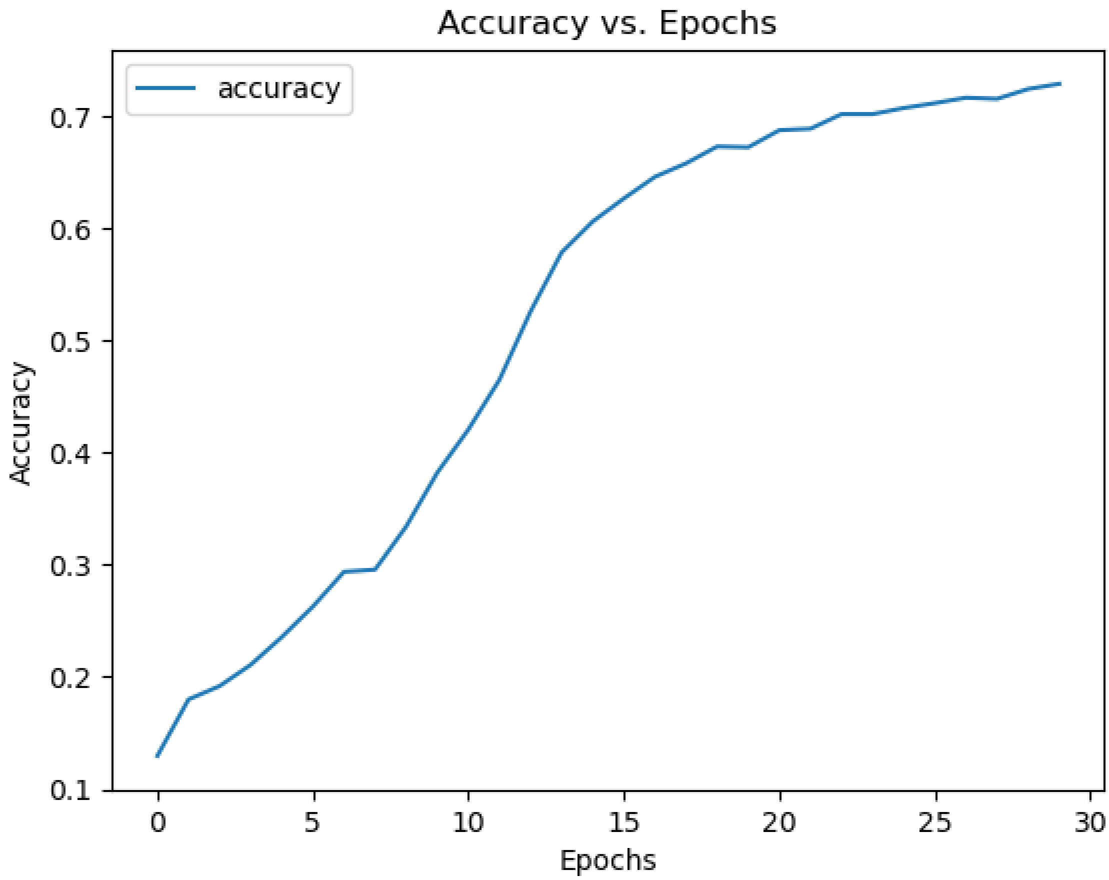
```
# Make a validation plot for the loss

loss_plot = df_mlp.plot(y='val_loss', title='Val_Loss vs. Epochs', legend='False')
loss_plot.set(xlabel='Epochs', ylabel='Val_Loss')
```

Out[ ]: [Text(0.5, 0, 'Epochs'), Text(0, 0.5, 'Val_Loss')]



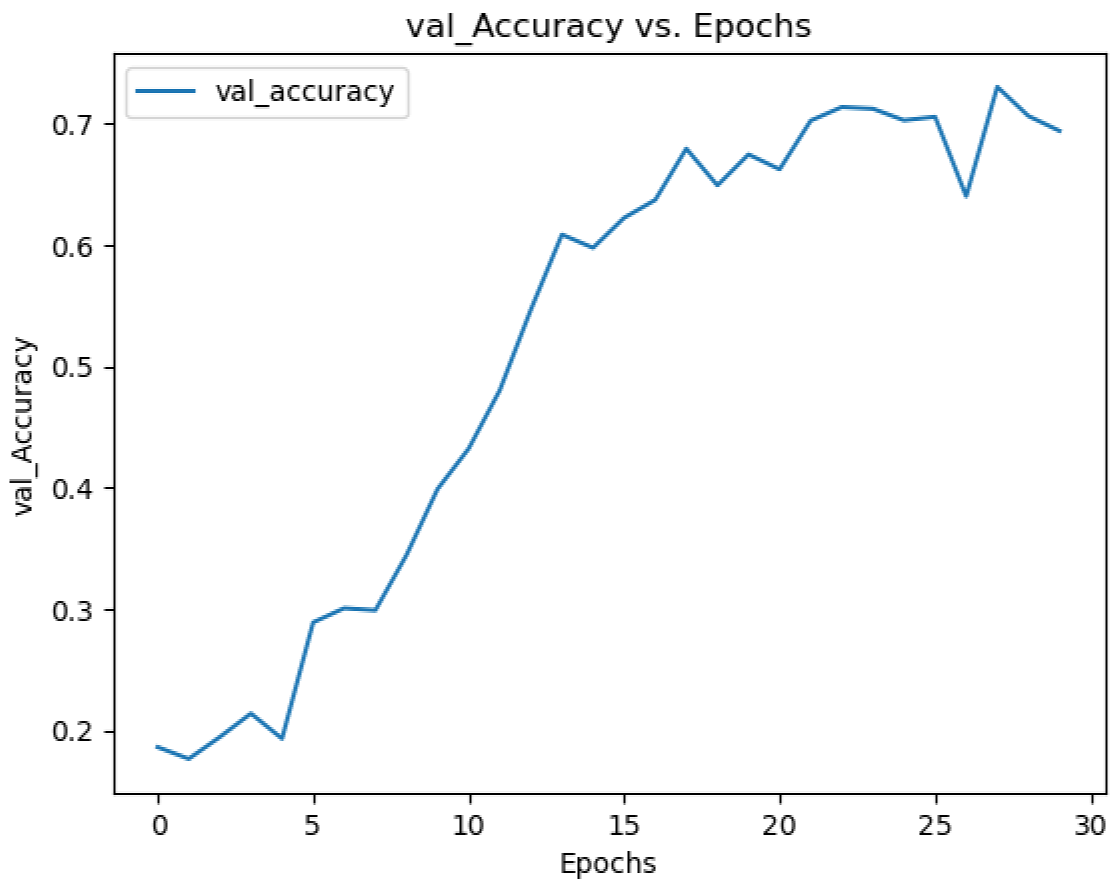
In [ ]:
```
# Make a plot for the accuracy

acc_plot = df_mlp.plot(y='accuracy', title='Accuracy vs. Epochs', legend='False')
acc_plot.set(xlabel='Epochs', ylabel='Accuracy')
```

Out[ ]:
```
[Text(0.5, 0, 'Epochs'), Text(0, 0.5, 'Accuracy')]
```



In [ ]:
```
# Make a plot for the validation accuracy

acc_plot = df_mlp.plot(y='val_accuracy', title='val_Accuracy vs. Epochs', legend='Fals
acc_plot.set(xlabel='Epochs', ylabel='val_Accuracy')
```

Out[ ]:
```
[Text(0.5, 0, 'Epochs'), Text(0, 0.5, 'val_Accuracy')]
```



3. CNN neural network classifier

- Build a CNN classifier model using the Sequential API. Your model should use the Conv2D, MaxPool2D, BatchNormalization, Flatten, Dense and Dropout layers. The final layer should again have a 10-way softmax output.
- You should design and build the model yourself. Feel free to experiment with different CNN architectures. *Hint: to achieve a reasonable accuracy you won't need to use more than 2 or 3 convolutional layers and 2 fully connected layers.)*
- The CNN model should use fewer trainable parameters than your MLP model.
- Compile and train the model (we recommend a maximum of 30 epochs), making use of both training and validation sets during the training run.
- Your model should track at least one appropriate metric, and use at least two callbacks during training, one of which should be a ModelCheckpoint callback.
- You should aim to beat the MLP model performance with fewer parameters!
- Plot the learning curves for loss vs epoch and accuracy vs epoch for both training and validation sets.
- Compute and display the loss and accuracy of the trained model on the test set.

In [ ]:
```
# Developing Sequential model for CNN classifier

model_cnn = Sequential([
    Conv2D(filters=16, input_shape=(32, 32, 1), kernel_size=(3, 3),
           activation='relu', name='conv_1'),
```

```
    BatchNormalization(),
    Dropout(0.3),
    Conv2D(filters=8, kernel_size=(3, 3), activation='relu', name='conv_2'),
    BatchNormalization(),
    MaxPooling2D(pool_size=(2, 2), name='pool_1'),
    Dropout(0.3),
    Flatten(name='flatten'),
    Dense(units=32, activation='relu', name='dense_1'),
    Dense(units=11, activation='softmax', name='dense_2')
])
```

In [ ]:
```
model_cnn.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv_1 (Conv2D)             (None, 30, 30, 16)        160


_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv_1 (Conv2D)             (None, 30, 30, 16)        160

 batch_normalization (BatchN  (None, 30, 30, 16)       64
 ormalization)

 dropout (Dropout)           (None, 30, 30, 16)        0

 conv_2 (Conv2D)             (None, 28, 28, 8)         1160

 batch_normalization_1 (Batc  (None, 28, 28, 8)        32
 hNormalization)

 pool_1 (MaxPooling2D)       (None, 14, 14, 8)         0

 dropout_1 (Dropout)         (None, 14, 14, 8)         0

 flatten (Flatten)           (None, 1568)              0

 dense_1 (Dense)             (None, 32)                50208

 dense_2 (Dense)             (None, 11)                363

=================================================================
Total params: 51,987
Trainable params: 51,939
Non-trainable params: 48
_________________________________________________________________
```

In [ ]:
```
model_cnn.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])
```

In [ ]:
```
checkpoint_path_cnn = 'model_cnn_checkpoints/checkpoint'
checkpoint_cnn = ModelCheckpoint(checkpoint_path_cnn, frequency='epoch', save_weights_
```

In [ ]:
```
history_cnn = model_cnn.fit(x_train, y_train, epochs=30,batch_size=264,validation_spli
```

```
Epoch 1/30
209/209 [==============================] - ETA: 0s - loss: 1.9576 - accuracy: 0.3279
Epoch 1: val_loss improved from inf to 1.30102, saving model to model_cnn_checkpoints
\checkpoint
209/209 [==============================] - 67s 312ms/step - loss: 1.9576 - accuracy:
0.3279 - val_loss: 1.3010 - val_accuracy: 0.5752
Epoch 2/30
209/209 [==============================] - ETA: 0s - loss: 0.9056 - accuracy: 0.7168
Epoch 2: val_loss improved from 1.30102 to 0.73070, saving model to model_cnn_checkpo
ints\checkpoint
209/209 [==============================] - 55s 265ms/step - loss: 0.9056 - accuracy:
0.7168 - val_loss: 0.7307 - val_accuracy: 0.7779
Epoch 3/30
209/209 [==============================] - ETA: 0s - loss: 0.6848 - accuracy: 0.7926
Epoch 3: val_loss improved from 0.73070 to 0.64180, saving model to model_cnn_checkpo
ints\checkpoint
209/209 [==============================] - 55s 263ms/step - loss: 0.6848 - accuracy:
0.7926 - val_loss: 0.6418 - val_accuracy: 0.8075
Epoch 4/30
209/209 [==============================] - ETA: 0s - loss: 0.6001 - accuracy: 0.8188
Epoch 4: val_loss improved from 0.64180 to 0.57969, saving model to model_cnn_checkpo
ints\checkpoint
209/209 [==============================] - 57s 273ms/step - loss: 0.6001 - accuracy:
0.8188 - val_loss: 0.5797 - val_accuracy: 0.8268
Epoch 5/30
209/209 [==============================] - ETA: 0s - loss: 0.5507 - accuracy: 0.8335
Epoch 5: val_loss improved from 0.57969 to 0.50648, saving model to model_cnn_checkpo
ints\checkpoint
209/209 [==============================] - 52s 249ms/step - loss: 0.5507 - accuracy:
0.8335 - val_loss: 0.5065 - val_accuracy: 0.8506
Epoch 6/30
208/209 [============================>.] - ETA: 0s - loss: 0.5165 - accuracy: 0.8422
Epoch 6: val_loss did not improve from 0.50648
209/209 [==============================] - 52s 249ms/step - loss: 0.5164 - accuracy:
0.8422 - val_loss: 0.5094 - val_accuracy: 0.8511
Epoch 7/30
209/209 [==============================] - ETA: 0s - loss: 0.4882 - accuracy: 0.8509
Epoch 7: val_loss did not improve from 0.50648
209/209 [==============================] - 52s 249ms/step - loss: 0.4882 - accuracy:
0.8509 - val_loss: 0.5436 - val_accuracy: 0.8386
Epoch 8/30
208/209 [============================>.] - ETA: 0s - loss: 0.4685 - accuracy: 0.8569
Epoch 8: val_loss did not improve from 0.50648
209/209 [==============================] - 52s 248ms/step - loss: 0.4687 - accuracy:
0.8568 - val_loss: 0.5106 - val_accuracy: 0.8464
Epoch 9/30
208/209 [============================>.] - ETA: 0s - loss: 0.4529 - accuracy: 0.8602
Epoch 9: val_loss improved from 0.50648 to 0.46371, saving model to model_cnn_checkpo
ints\checkpoint
209/209 [==============================] - 52s 248ms/step - loss: 0.4528 - accuracy:
0.8603 - val_loss: 0.4637 - val_accuracy: 0.8634
Epoch 10/30
208/209 [============================>.] - ETA: 0s - loss: 0.4445 - accuracy: 0.8629
Epoch 10: val_loss did not improve from 0.46371
209/209 [==============================] - 52s 246ms/step - loss: 0.4445 - accuracy:
0.8629 - val_loss: 0.5312 - val_accuracy: 0.8389
Epoch 11/30
209/209 [==============================] - ETA: 0s - loss: 0.4314 - accuracy: 0.8688
Epoch 11: val_loss improved from 0.46371 to 0.45768, saving model to model_cnn_checkp
oints\checkpoint
```

```
209/209 [==============================] - 52s 247ms/step - loss: 0.4314 - accuracy:
0.8688 - val_loss: 0.4577 - val_accuracy: 0.8647
Epoch 12/30
208/209 [============================>.] - ETA: 0s - loss: 0.4204 - accuracy: 0.8695
Epoch 12: val_loss did not improve from 0.45768
209/209 [==============================] - 53s 252ms/step - loss: 0.4204 - accuracy:
0.8695 - val_loss: 0.4636 - val_accuracy: 0.8645
Epoch 13/30
209/209 [==============================] - ETA: 0s - loss: 0.4087 - accuracy: 0.8748
Epoch 13: val_loss improved from 0.45768 to 0.43977, saving model to model_cnn_checkp
oints\checkpoint
209/209 [==============================] - 1192s 6s/step - loss: 0.4087 - accuracy:
0.8748 - val_loss: 0.4398 - val_accuracy: 0.8726
Epoch 14/30
209/209 [==============================] - ETA: 0s - loss: 0.4034 - accuracy: 0.8755
Epoch 14: val_loss improved from 0.43977 to 0.42802, saving model to model_cnn_checkp
oints\checkpoint
209/209 [==============================] - 60s 285ms/step - loss: 0.4034 - accuracy:
0.8755 - val_loss: 0.4280 - val_accuracy: 0.8738
Epoch 15/30
209/209 [==============================] - ETA: 0s - loss: 0.3941 - accuracy: 0.8774
Epoch 15: val_loss did not improve from 0.42802
209/209 [==============================] - 60s 288ms/step - loss: 0.3941 - accuracy:
0.8774 - val_loss: 0.4876 - val_accuracy: 0.8529
Epoch 16/30
209/209 [==============================] - ETA: 0s - loss: 0.3864 - accuracy: 0.8798
Epoch 16: val_loss did not improve from 0.42802
209/209 [==============================] - 59s 284ms/step - loss: 0.3864 - accuracy:
0.8798 - val_loss: 0.4543 - val_accuracy: 0.8654
Epoch 17/30
208/209 [============================>.] - ETA: 0s - loss: 0.3832 - accuracy: 0.8817
Epoch 17: val_loss did not improve from 0.42802
209/209 [==============================] - 51s 245ms/step - loss: 0.3832 - accuracy:
0.8817 - val_loss: 0.5025 - val_accuracy: 0.8523
Epoch 18/30
209/209 [==============================] - ETA: 0s - loss: 0.3756 - accuracy: 0.8834
Epoch 18: val_loss improved from 0.42802 to 0.42650, saving model to model_cnn_checkp
oints\checkpoint
209/209 [==============================] - 55s 264ms/step - loss: 0.3756 - accuracy:
0.8834 - val_loss: 0.4265 - val_accuracy: 0.8789
Epoch 19/30
209/209 [==============================] - ETA: 0s - loss: 0.3739 - accuracy: 0.8836
Epoch 19: val_loss did not improve from 0.42650
209/209 [==============================] - 55s 264ms/step - loss: 0.3739 - accuracy:
0.8836 - val_loss: 0.4295 - val_accuracy: 0.8739
Epoch 20/30
209/209 [==============================] - ETA: 0s - loss: 0.3688 - accuracy: 0.8854
Epoch 20: val_loss improved from 0.42650 to 0.40848, saving model to model_cnn_checkp
oints\checkpoint
209/209 [==============================] - 55s 262ms/step - loss: 0.3688 - accuracy:
0.8854 - val_loss: 0.4085 - val_accuracy: 0.8814
Epoch 21/30
209/209 [==============================] - ETA: 0s - loss: 0.3620 - accuracy: 0.8867
Epoch 21: val_loss did not improve from 0.40848
209/209 [==============================] - 55s 261ms/step - loss: 0.3620 - accuracy:
0.8867 - val_loss: 0.4499 - val_accuracy: 0.8661
Epoch 22/30
209/209 [==============================] - ETA: 0s - loss: 0.3564 - accuracy: 0.8893
Epoch 22: val_loss did not improve from 0.40848
209/209 [==============================] - 60s 286ms/step - loss: 0.3564 - accuracy:
```

```
0.8893 - val_loss: 0.4331 - val_accuracy: 0.8730
Epoch 23/30
209/209 [==============================] - ETA: 0s - loss: 0.3529 - accuracy: 0.8899
Epoch 23: val_loss did not improve from 0.40848
209/209 [==============================] - 59s 282ms/step - loss: 0.3529 - accuracy:
0.8899 - val_loss: 0.4581 - val_accuracy: 0.8662
Epoch 24/30
209/209 [==============================] - ETA: 0s - loss: 0.3516 - accuracy: 0.8905
Epoch 24: val_loss did not improve from 0.40848
209/209 [==============================] - 59s 284ms/step - loss: 0.3516 - accuracy:
0.8905 - val_loss: 0.4241 - val_accuracy: 0.8757
Epoch 25/30
209/209 [==============================] - ETA: 0s - loss: 0.3487 - accuracy: 0.8907
Epoch 25: val_loss did not improve from 0.40848
209/209 [==============================] - 58s 277ms/step - loss: 0.3487 - accuracy:
0.8907 - val_loss: 0.4488 - val_accuracy: 0.8656
Epoch 26/30
209/209 [==============================] - ETA: 0s - loss: 0.3479 - accuracy: 0.8912
Epoch 26: val_loss improved from 0.40848 to 0.40549, saving model to model_cnn_checkp
oints\checkpoint
209/209 [==============================] - 58s 279ms/step - loss: 0.3479 - accuracy:
0.8912 - val_loss: 0.4055 - val_accuracy: 0.8821
Epoch 27/30
209/209 [==============================] - ETA: 0s - loss: 0.3428 - accuracy: 0.8927
Epoch 27: val_loss did not improve from 0.40549
209/209 [==============================] - 62s 299ms/step - loss: 0.3428 - accuracy:
0.8927 - val_loss: 0.4278 - val_accuracy: 0.8758
Epoch 28/30
209/209 [==============================] - ETA: 0s - loss: 0.3409 - accuracy: 0.8939
Epoch 28: val_loss did not improve from 0.40549
209/209 [==============================] - 58s 277ms/step - loss: 0.3409 - accuracy:
0.8939 - val_loss: 0.4152 - val_accuracy: 0.8783
Epoch 29/30
209/209 [==============================] - ETA: 0s - loss: 0.3411 - accuracy: 0.8937
Epoch 29: val_loss did not improve from 0.40549
209/209 [==============================] - 58s 280ms/step - loss: 0.3411 - accuracy:
0.8937 - val_loss: 0.4272 - val_accuracy: 0.8750
Epoch 30/30
208/209 [============================>.] - ETA: 0s - loss: 0.3377 - accuracy: 0.8934
Epoch 30: val_loss did not improve from 0.40549
209/209 [==============================] - 55s 261ms/step - loss: 0.3378 - accuracy:
0.8934 - val_loss: 0.4138 - val_accuracy: 0.8797
```

In [ ]:
```python
df_cnn = pd.DataFrame(history_cnn.history)
df_cnn.head()
```

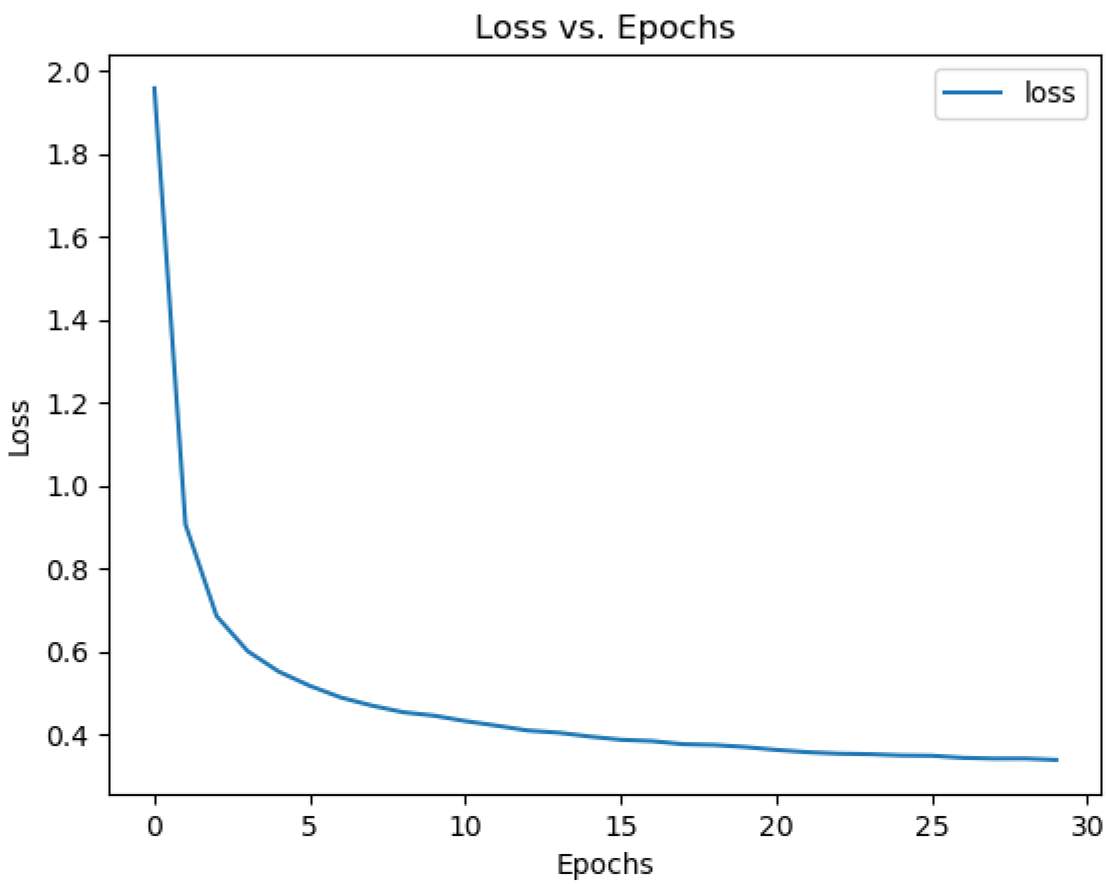
Out[ ]:

	loss	accuracy	val_loss	val_accuracy
0	1.957572	0.327873	1.301021	0.575157
1	0.905590	0.716847	0.730696	0.777942
2	0.684754	0.792581	0.641799	0.807535
3	0.600095	0.818827	0.579692	0.826809
4	0.550703	0.833534	0.506482	0.850560

In [ ]:
```python
# Make a plot for the loss
loss_plot = df_cnn.plot(y='loss', title='Loss vs. Epochs', legend='False')
```

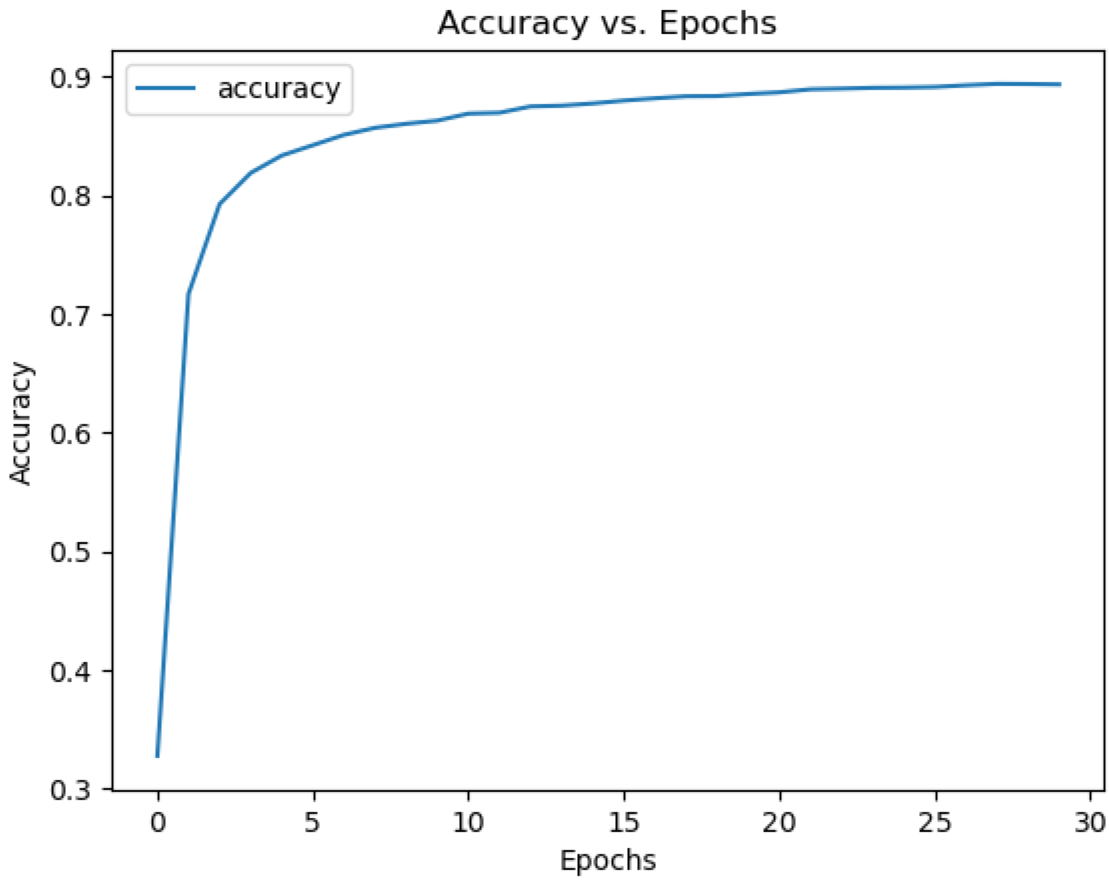
```
loss_plot.set(xlabel='Epochs', ylabel='Loss')
```

Out[ ]: [Text(0.5, 0, 'Epochs'), Text(0, 0.5, 'Loss')]



In [ ]:
```
# Make a plot for the accuracy

acc_plot = df_cnn.plot(y='accuracy', title='Accuracy vs. Epochs', legend='False')
acc_plot.set(xlabel='Epochs', ylabel='Accuracy')
```

Out[ ]: [Text(0.5, 0, 'Epochs'), Text(0, 0.5, 'Accuracy')]



4. Get model predictions

- Load the best weights for the MLP and CNN models that you saved during the training run.
- Randomly select 5 images and corresponding labels from the test set and display the images with their labels.
- Alongside the image and label, show each model's predictive distribution as a bar chart, and the final model prediction given by the label with maximum probability.

In [ ]:

```
# Loading saved best weights

model_mlp.load_weights(checkpoint_path_mlp)

model_cnn.load_weights(checkpoint_path_cnn)
```

Out[ ]:

```
<tensorflow.python.checkpoint.checkpoint.CheckpointLoadStatus at 0x1c501e39bc8>
```

In [ ]:

```
# Evaluating MLP model with saved best weights on test data
model_mlp.evaluate(x_test,y_test)
```

```
814/814 [==============================] - 3s 4ms/step - loss: 1.0442 - accuracy: 0.6
942
```

Out[ ]:

```
[1.0441722869873047, 0.6942225098609924]
```

In [ ]:

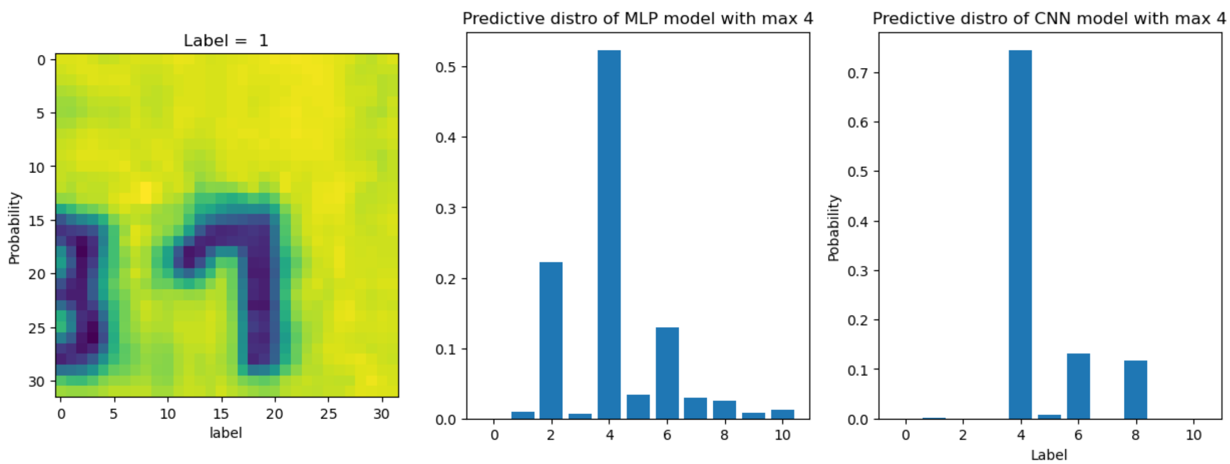
```
# Evaluating CNN model with saved best weights on test data
model_cnn.evaluate(x_test,y_test)
```

```
814/814 [==============================] - 8s 10ms/step - loss: 0.4688 - accuracy: 0.
8635
```

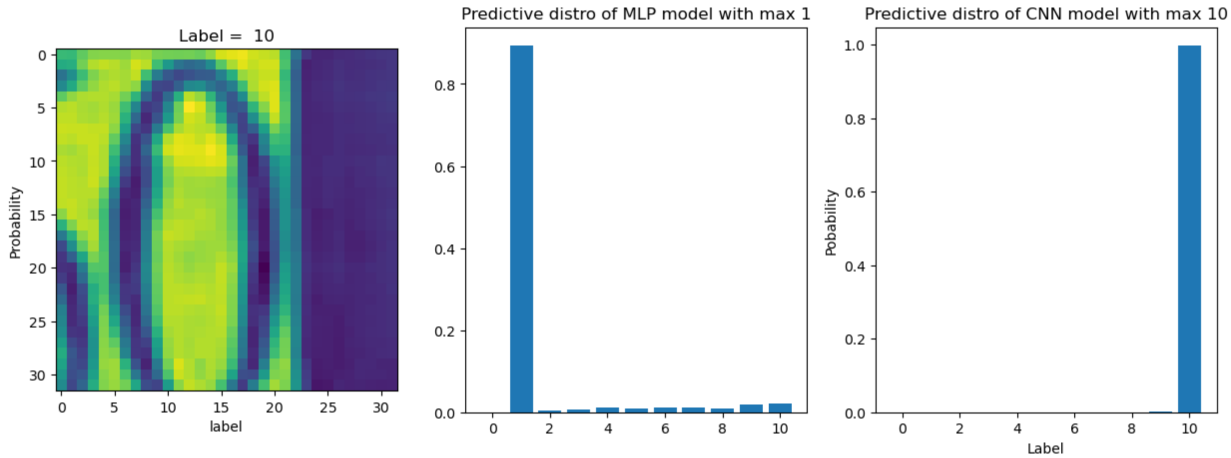
Out[ ]:
```
[0.4688352644443512, 0.8634757399559021]
```

In [ ]:
```
# Pltotting the image and the predictive distribution bar charts for random images in
prediction_mlp = []
prediction_cnn = []
for i in range(0,5):
    j=int(np.random.random()*100)
    prediction_mlp = model_mlp.predict(x_test[j][np.newaxis,...])
    prediction_cnn = model_cnn.predict(x_test[j][np.newaxis,...])

    fig, axs = plt.subplots(1, 3, figsize=(15, 5))

    axs[0].set_title('Label =  %i' %((y_test[j])))
    axs[0].imshow(x_test[j])

    axs[1].set_title('Predictive distro of MLP model with max %i' %(np.argmax(predicti
    axs[1].bar(range(0,11), prediction_mlp[0])
    axs[0].set_xlabel('label')
    axs[0].set_ylabel('Probability')

    axs[2].set_title('Predictive distro of CNN model with max %i' %(np.argmax(predicti
    axs[2].bar(range(0,11), prediction_cnn[0])
    axs[2].set_xlabel('Label')
    axs[2].set_ylabel('Pobability')
    plt.show()
```

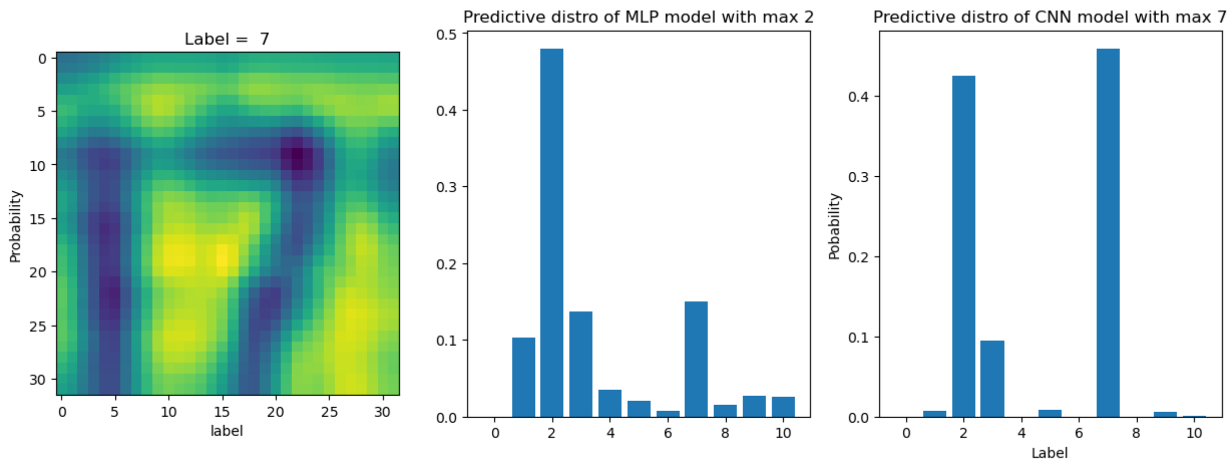
```
1/1 [==============================] - 0s 15ms/step
1/1 [==============================] - 0s 15ms/step
```



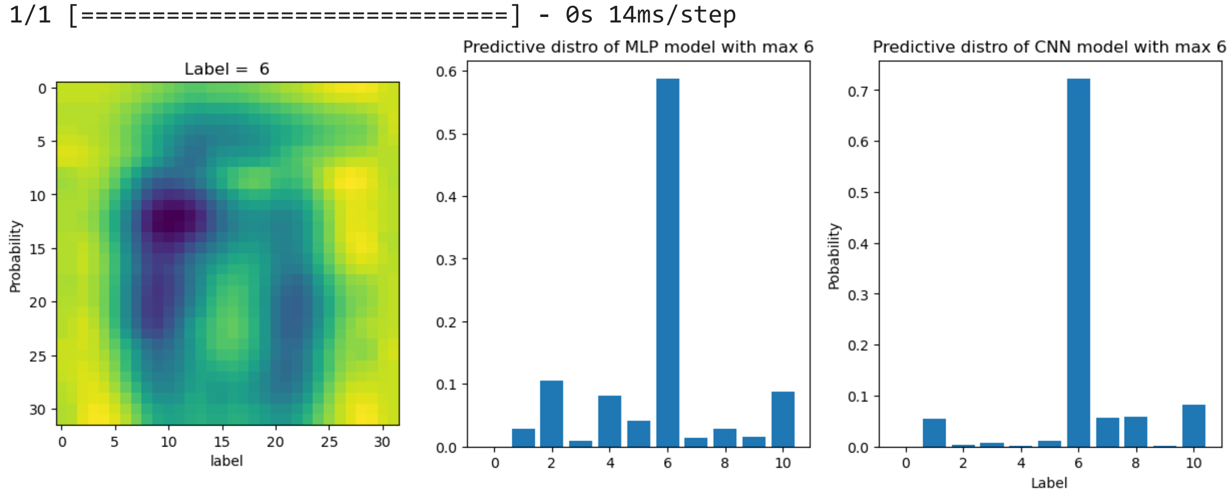

```
1/1 [==============================] - 0s 29ms/step
1/1 [==============================] - 0s 13ms/step
```



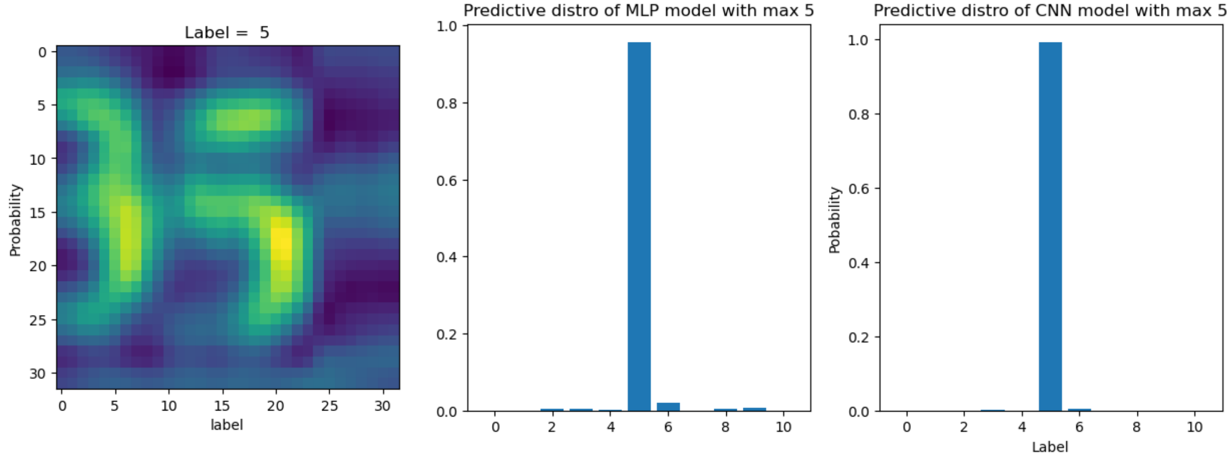

```
1/1 [==============================] - 0s 15ms/step
1/1 [==============================] - 0s 14ms/step
```




```
1/1 [==============================] - 0s 15ms/step
1/1 [==============================] - 0s 14ms/step
```




```
1/1 [==============================] - 0s 15ms/step
1/1 [==============================] - 0s 14ms/step
```


Label = 5
Probability
label
Predictive distro of MLP model with max 5
Predictive distro of CNN model with max 5
Pobability
Label

In [ ]: