

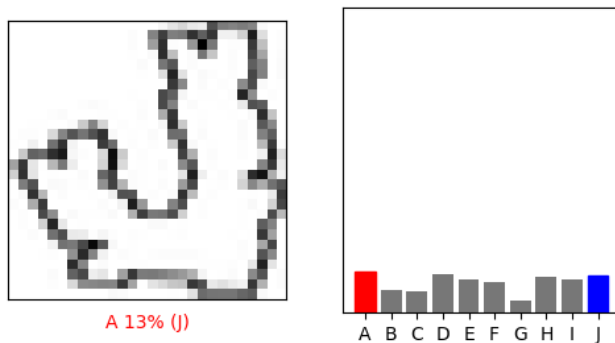
CPSC 501- Assignment 2 Part 2: MNIST Logistic Regression on a replacement for MNIST dataset

Partial Model:

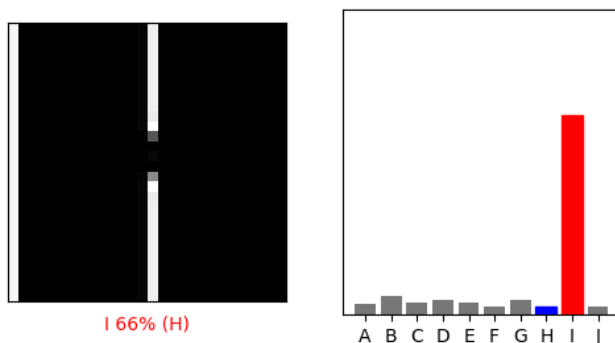
- I chose three layer model like in part 1. I added an extra layer of `'tf.keras.layers.Dense(128, activation='relu')'`. I chose the activation function as ReLu because it worked very well from part 1. I changed last layer's activation function to 'Softmax' like I did in part 1 of the assignment. Also, I changed optimizer to 'Adam' and loss function to 'sparse_categorical_crossentropy'. Lastly, I changed epochs from 1 to 5. Increasing epochs in this model did not increase accuracy of the model so I kept it at 5. This partial model gave me a train accuracy of 90.3% and test accuracy of 93.3%.

Three Images and their original Partial model label:

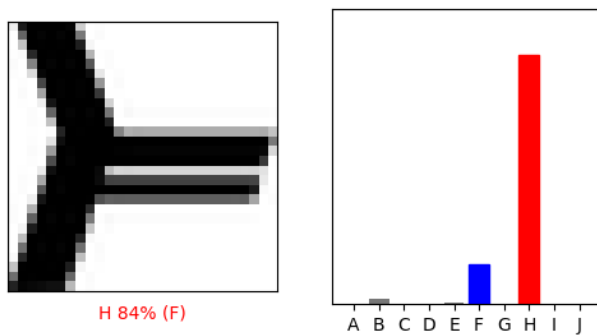
- The partial model was unable to correctly classify the above following of an 'J', where it predicted it was a 'A' with 13% confidence.



- The partial model was unable to correctly classify the following image of a 'H', where it predicted it was an 'I' with 66% confidence.



- The partial model was unable to correctly classify the following image of a 'F', where it predicted it was a 'H' with 84% confidence.

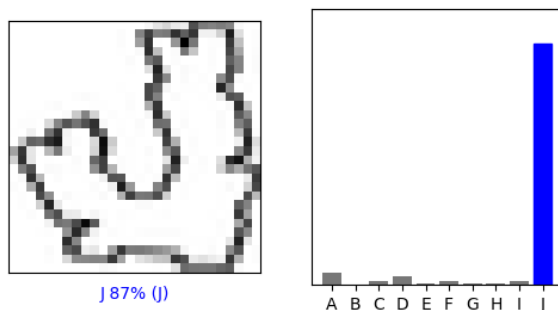


Changes to the model and improvement:

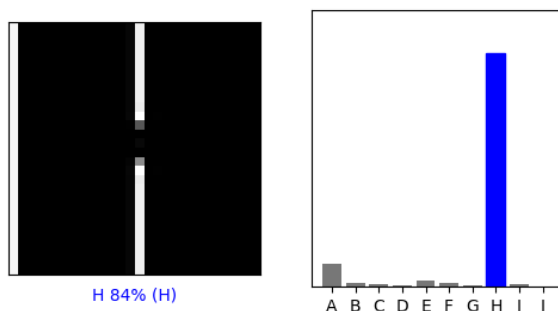
- I added Reshape layer that reshapes inputs into the given shape.
- I added 2D Convolution Layer, this layer creates a convolution kernel that is convolved with layers input which helps produce a tensor of outputs
- I added MaxPooling2D layer which downsamples the input along its spatial dimensions by taking the maximum value over an input window for each channel of the input.
- I added Dropout layer which randomly sets input units to 0 with a frequency of rate at each step during training time, which helps prevent overfitting.
- All these layers added helped the model to achieve higher prediction accuracy of 95.4% on test data.

Three Images and their Complete model label:

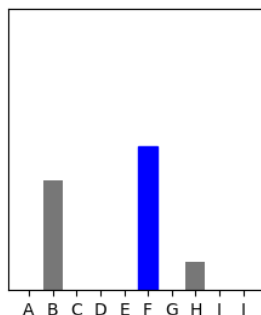
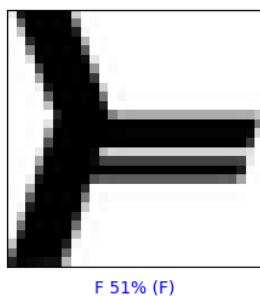
- The complete model was able to correctly classify the following image of a 'J' with 87% confidence.



- The complete model was able to correctly classify the following image of a 'H' with 84% confidence.



- The complete model was able to correctly classify the following image of a 'F' with 51% confidence.



Accuracy:

Train accuracy: 99.3%

Test Accuracy: 95.3%

notMNIST-Starter.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Files

sample_data

notMNIST-Complete.h5

notMNIST.npz

+ Code + Text

```

469/469 - 29s - loss: 0.1526 - accuracy: 0.9504 - 29s/epoch - 62ms/step
Epoch 13/20
469/469 - 29s - loss: 0.1323 - accuracy: 0.9564 - 29s/epoch - 62ms/step
Epoch 14/20
469/469 - 29s - loss: 0.1231 - accuracy: 0.9600 - 29s/epoch - 62ms/step
Epoch 15/20
469/469 - 29s - loss: 0.1107 - accuracy: 0.9638 - 29s/epoch - 62ms/step
Epoch 16/20
469/469 - 30s - loss: 0.1023 - accuracy: 0.9666 - 30s/epoch - 64ms/step
Epoch 17/20
469/469 - 29s - loss: 0.0919 - accuracy: 0.9704 - 29s/epoch - 62ms/step
Epoch 18/20
469/469 - 29s - loss: 0.0879 - accuracy: 0.9715 - 29s/epoch - 61ms/step
Epoch 19/20
469/469 - 29s - loss: 0.0826 - accuracy: 0.9731 - 29s/epoch - 61ms/step
Epoch 20/20
469/469 - 29s - loss: 0.0768 - accuracy: 0.9750 - 29s/epoch - 62ms/step
<keras.callbacks.History at 0x7f5bd46d3a10>

[85] print("--Evaluate model--")
model_loss1, model_acc1 = model.evaluate(x_train, y_train, verbose=2)
model_loss2, model_acc2 = model.evaluate(x_test, y_test, verbose=2)
print(f"Train / Test Accuracy: {model_acc1*100:.1f}% / {model_acc2*100:.1f}%")

--Evaluate model--
1875/1875 - 12s - loss: 0.0335 - accuracy: 0.9926 - 12s/epoch - 6ms/step
313/313 - 2s - loss: 0.2247 - accuracy: 0.9527 - 2s/epoch - 7ms/step
Train / Test Accuracy: 99.3% / 95.3%

model.save("notMNIST-Complete.h5")

```

85.11 GB available

0s completed at 4:42 PM