Handwriting Recognition

Deep Learning

Lydia Jin, Xi Qian, Yuyang Wang

Dataset: IAM Dataset

Forms_for_parsing.txt

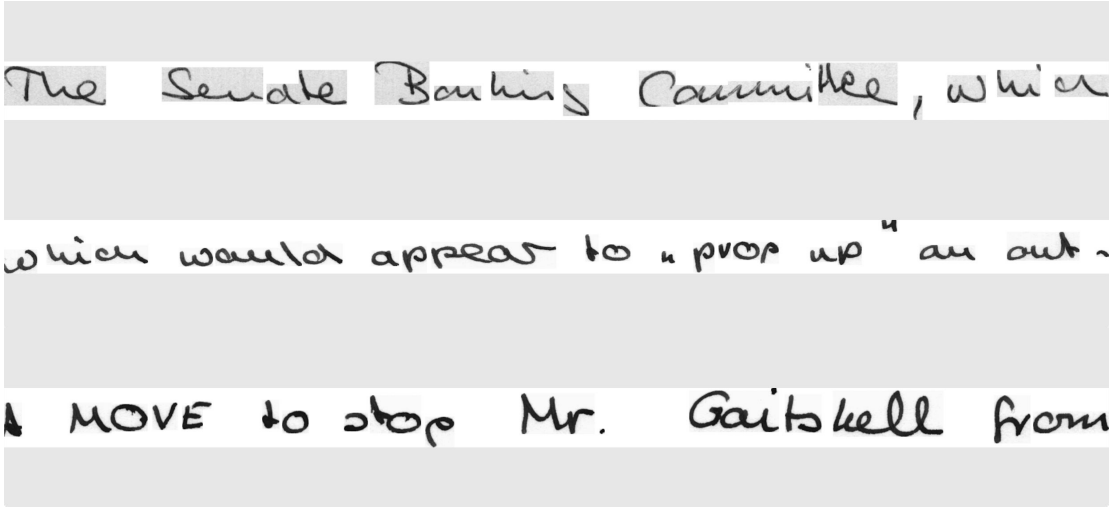
```
a01-003 002
a01-003u 000
a01-003x 003
a01-007 004
a01-007u 000
a01-007x 003
a01-011 005
a01-011u 000
a01-011x 006
a01-014 007
```

Image dataset

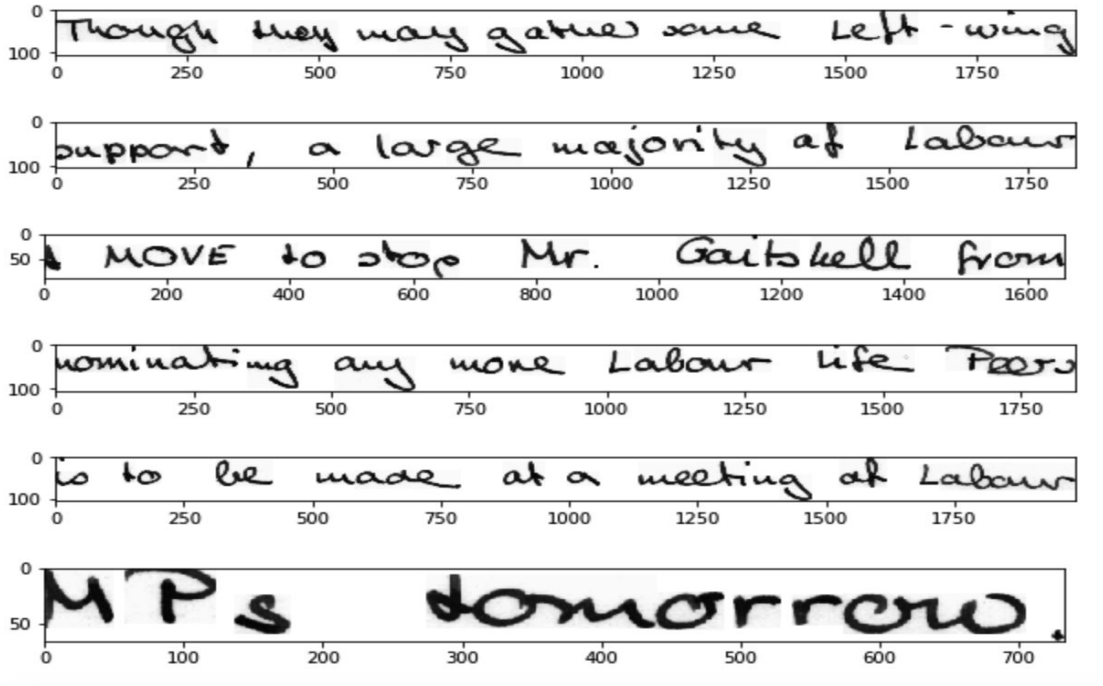
4899 number of images

50+ writers

Image filename: a01-000u-s00-03.png



Example of Handwritten scripts



Data Pre-Processing

1. Writer number and image file mapping

```
{'a01-000u': '000',
 'a01-000x': '001',
 'a01-003': '002',
 'a01-003u': '000',
```

```
1 img_targets
```

```
array(['000', '000'
```

2. Label Encode: [‘001’] -> [‘1’]
3. Train, Validation, Test split: 7:1.5:1.5

```
(3429,) (735,) (735,)
(3429,) (735,) (735,)
```

Data Pre-Processing

4. Resize and crop (generator function)

(1) Resize image-Keep same aspect ratio
(2) Crop 113*113
(3) Keep 10% of images

5. Normalize pixel value (0-255) →(0-1)
6. One hot encoding Y (to_categorical)

Keras Vs.Pytorch

```
model = Sequential()
model.add(Conv2D(32, (3, 3), activation='relu', input_shape=
(32, 32, 3)))
model.add(MaxPool2D())
model.add(Conv2D(16, (3, 3), activation='relu'))
model.add(MaxPool2D())
model.add(Flatten())
model.add(Dense(10, activation='softmax'))
```

- Enabling GPU acceleration is handled implicitly in Keras
- More readable and concise
- Skipping the implementational details

```
class Net(nn.Module):
    def __init__(self):
        super(Net, self).__init__()

        self.conv1 = nn.Conv2d(3, 32, 3)
        self.conv2 = nn.Conv2d(32, 16, 3)
        self.fc1 = nn.Linear(16 * 6 * 6, 10)
        self.pool = nn.MaxPool2d(2, 2)

    def forward(self, x):
        x = self.pool(F.relu(self.conv1(x)))
        x = self.pool(F.relu(self.conv2(x)))
        x = x.view(-1, 16 * 6 * 6)
        x = F.log_softmax(self.fc1(x), dim=-1)

        return x
```

- PyTorch requires us to specify when to transfer data between the CPU and GPU

Network Construction

Sequential Vs. Functional API

```
model = Sequential()
```

Model type

output channels

```
model.add(Convolution2D(filters=32, kernel_size=(3, 3), strides=(2, 2), padding='same', name='conv1'))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2), strides=(2, 2), name='pool1'))
```

Network Construction

```
model.add(Convolution2D(filters=32, kernel_size=(3, 3), strides=(2, 2), padding='same', name='conv1'))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2), strides=(2, 2), name='pool1'))

model.add(Convolution2D(filters=64, kernel_size=(3, 3), strides=(1, 1), padding='same', name='conv2'))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2), strides=(2, 2), name='pool2'))

model.add(Convolution2D(filters=128, kernel_size=(3, 3), strides=(1, 1), padding='same', name='conv3'))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2), strides=(2, 2), name='pool3'))
```

Network Construction

```
model.add(Flatten())
model.add(Dense(512, name='dense1'))
model.add(Activation('relu'))
model.add(Dropout(0.4))
```

flatten the output to enter fully connected layers

```
model.compile(loss='categorical_crossentropy', optimizer=Adam(), metrics=['accuracy'])
```

Parameter Tuning

- Number of layers
- Kernel size
- Optimizer

3-Layer CNN (Adam) Model Output

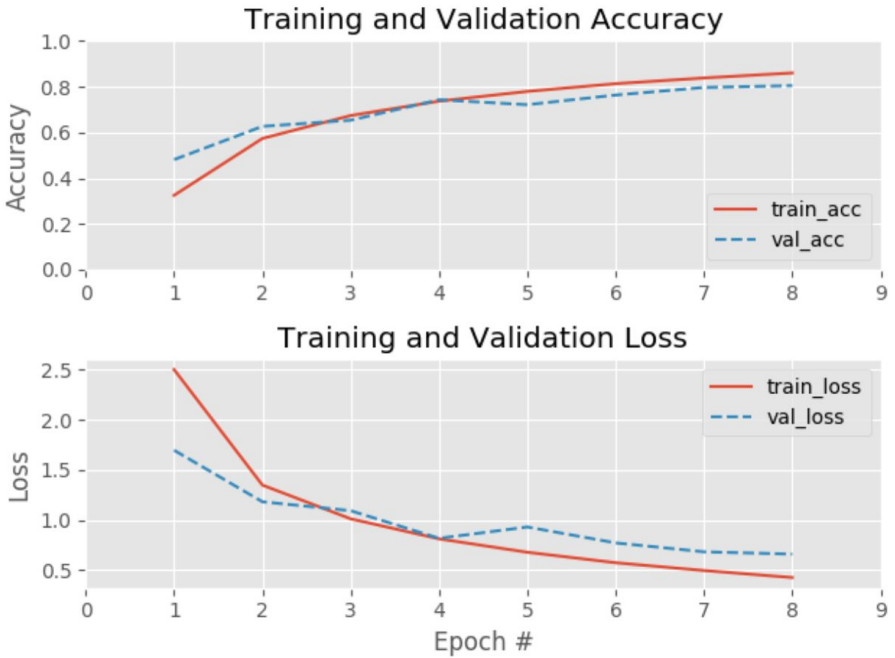
```
Epoch 1/8
1000/1000 [==============================] - 2065s 2s/step - loss: 2.5778 - acc: 0.3068 - val_loss: 1.6233 - val_acc: 0.5028

Epoch 00001: saving model to check-01-1.6233.hdf5
Epoch 2/8
1000/1000 [==============================] - 1813s 2s/step - loss: 1.4534 - acc: 0.5520 - val_loss: 1.0423 - val_acc: 0.6724

Epoch 00002: saving model to check-02-1.0423.hdf5
Epoch 3/8
1000/1000 [==============================] - 1799s 2s/step - loss: 1.0960 - acc: 0.6599 - val_loss: 0.8940 - val_acc: 0.7204

Epoch 00003: saving model to check-03-0.8940.hdf5
Epoch 4/8
1000/1000 [==============================] - 1794s 2s/step - loss: 0.9058 - acc: 0.7189 - val_loss: 0.7573 - val_acc: 0.7642

Epoch 00004: saving model to check-04-0.7573.hdf5
Epoch 5/8
1000/1000 [==============================] - 1789s 2s/step - loss: 0.7845 - acc: 0.7559 - val_loss: 0.6939 - val_acc: 0.7839

Epoch 00005: saving model to check-05-0.6939.hdf5
Epoch 6/8
1000/1000 [==============================] - 1807s 2s/step - loss: 0.6902 - acc: 0.7852 - val_loss: 0.6555 - val_acc: 0.7984

Epoch 00006: saving model to check-06-0.6555.hdf5
Epoch 7/8
1000/1000 [==============================] - 1797s 2s/step - loss: 0.6219 - acc: 0.8066 - val_loss: 0.5794 - val_acc: 0.8222

Epoch 00007: saving model to check-07-0.5794.hdf5
Epoch 8/8
1000/1000 [==============================] - 1836s 2s/step - loss: 0.5696 - acc: 0.8222 - val_loss: 0.5786 - val_acc: 0.8194
```

```
('Accuracy = ', 0.81283999999999999)
```

model.fit_generator output

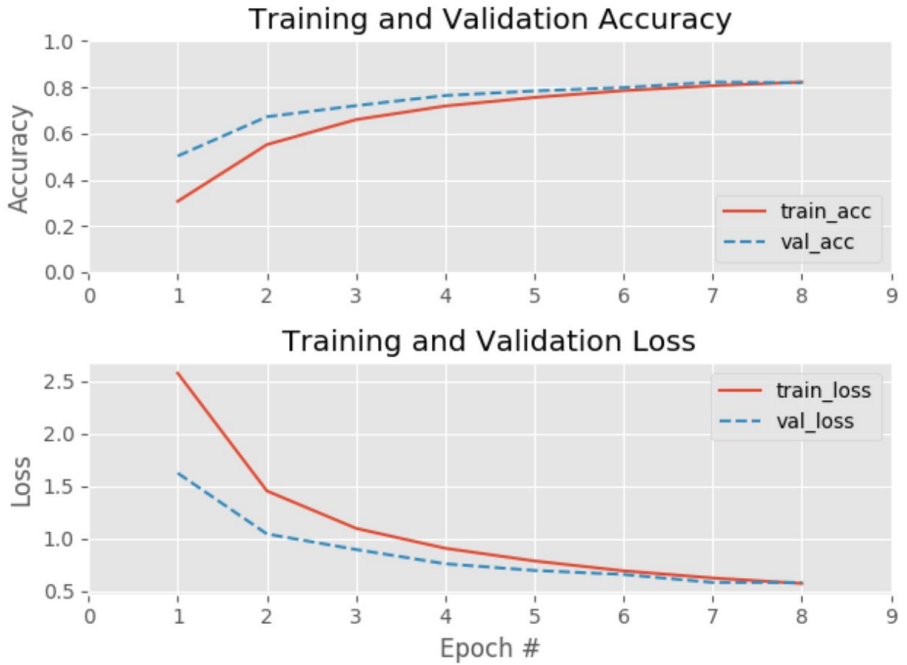
Training and Validation Accuracy/Loss Plots

No Dropout Layers

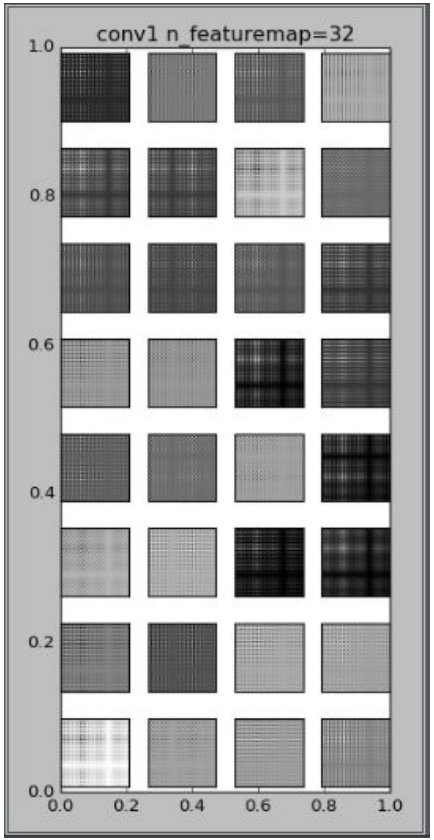


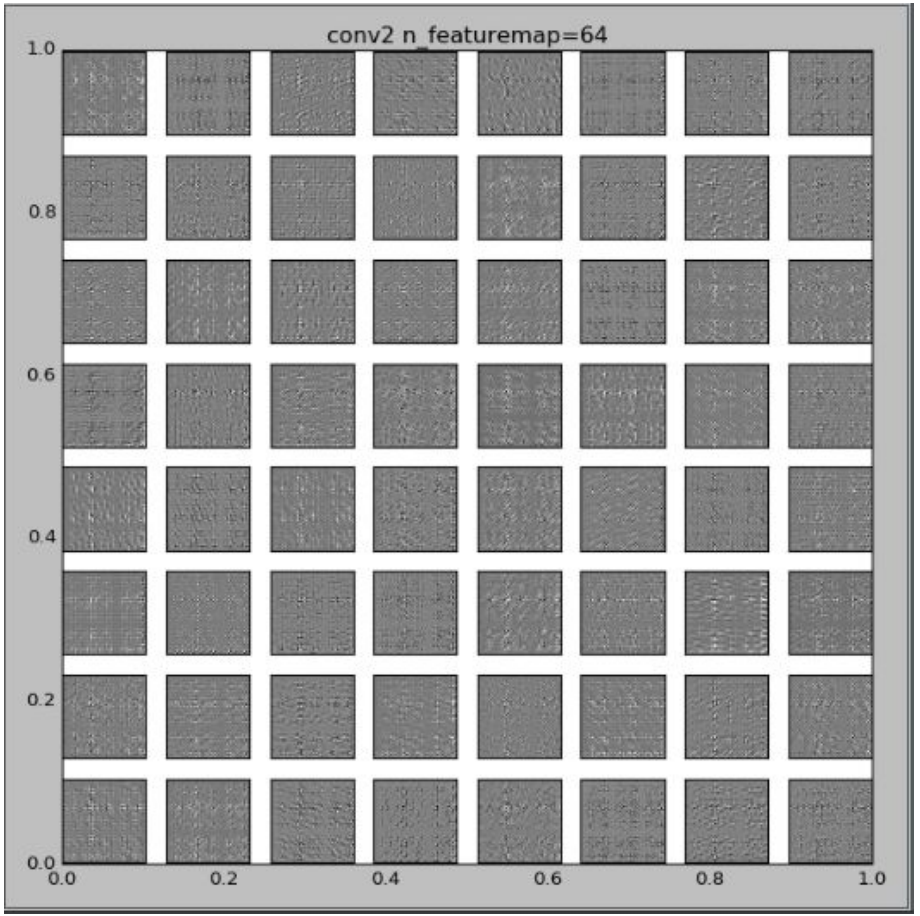
Training and Validation Accuracy/Loss Plots

Dropout Layers Included

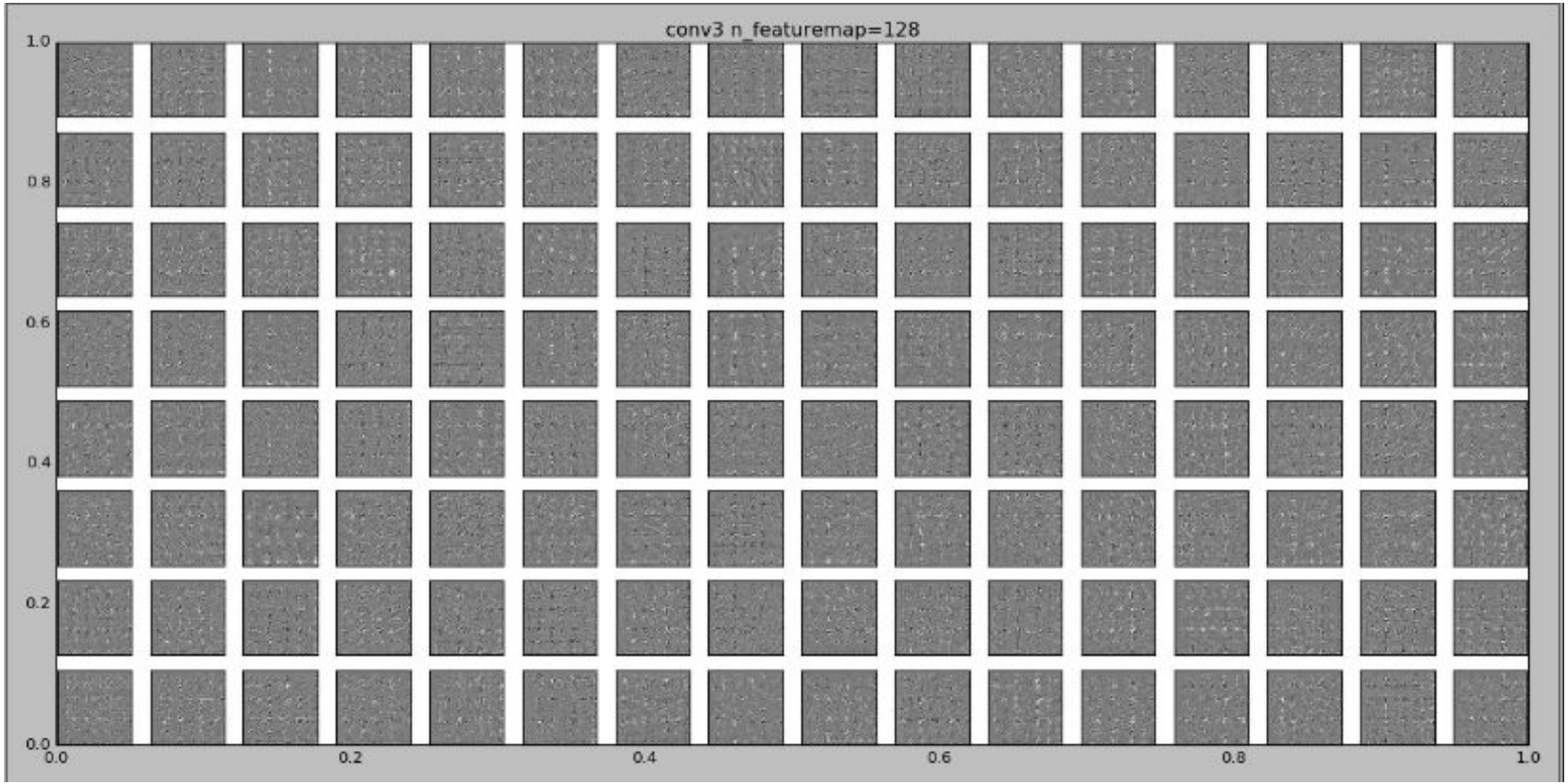


Visualizing CNN Filters

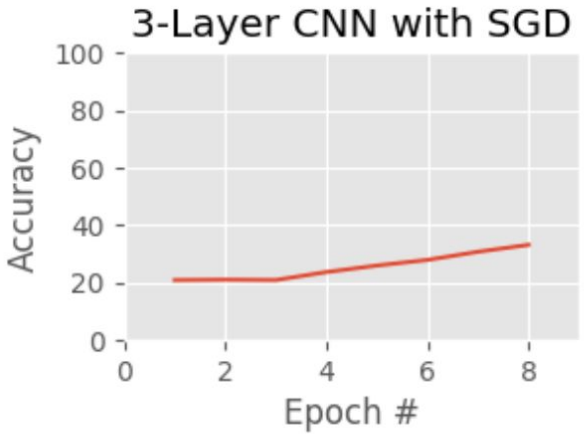


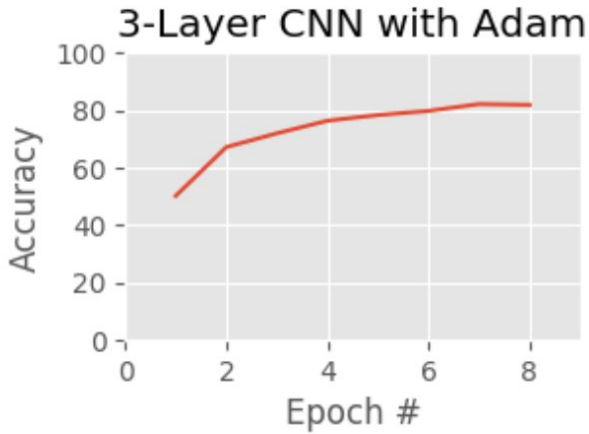


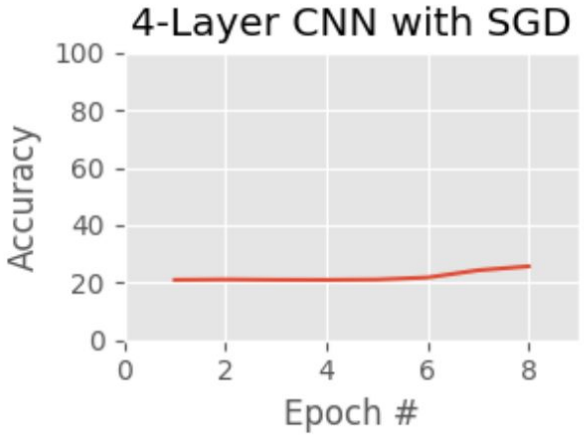
Visualizing CNN Filters (cont.)

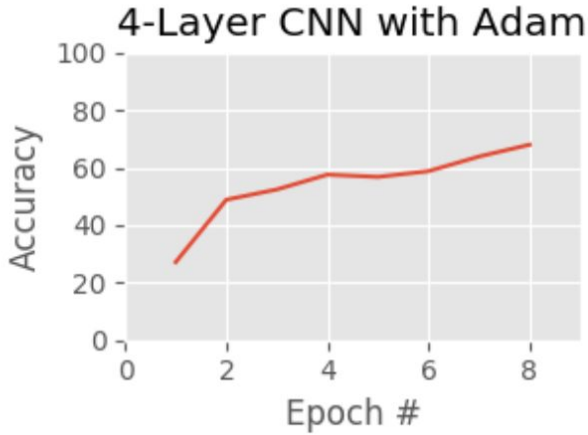


Validation Accuracy









Summary and Conclusion

Best model:
- Adam optimizer
- 3-layer CNN

Learning process:
- Keras
- CNN

Improvements:
- Further visualization (e.g., confusion matrix, gradients)
- Parameters (e.g., # epochs)

Questions?

References

Dataset: https://www.kaggle.com/tejasreddy/iam-handwriting-top50
https://www.learnopencv.com/image-classification-using-convolutional-neural-networks-in-keras/
Figure 1:
https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2
Checkpoint code: https://machinelearningmastery.com/check-point-deep-learning-models-keras/
Generator code: https://www.kaggle.com/tejasreddy/offline-handwriting-recognition-cnn/notebook
CNN Background Information:
https://www.learnopencv.com/image-classification-using-convolutional-neural-networks-in-keras/
Save/Load Keras models: http://faroit.com/keras-docs/2.0.2/models/about-keras-models/
CNN Filter code: https://fairyonice.github.io/Visualization%20of%20Filters%20with%20Keras.html