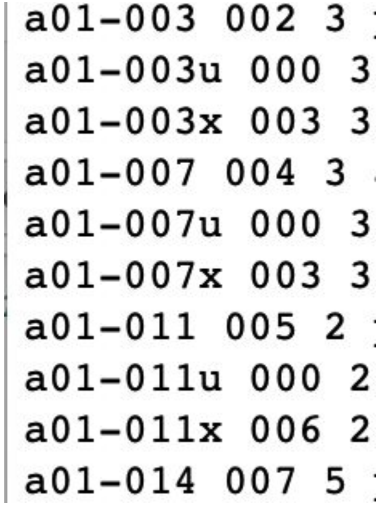
Image Classification: IAM Handwriting

Lydia Jin, Xi Qian, Yuyang Wang

Dataset

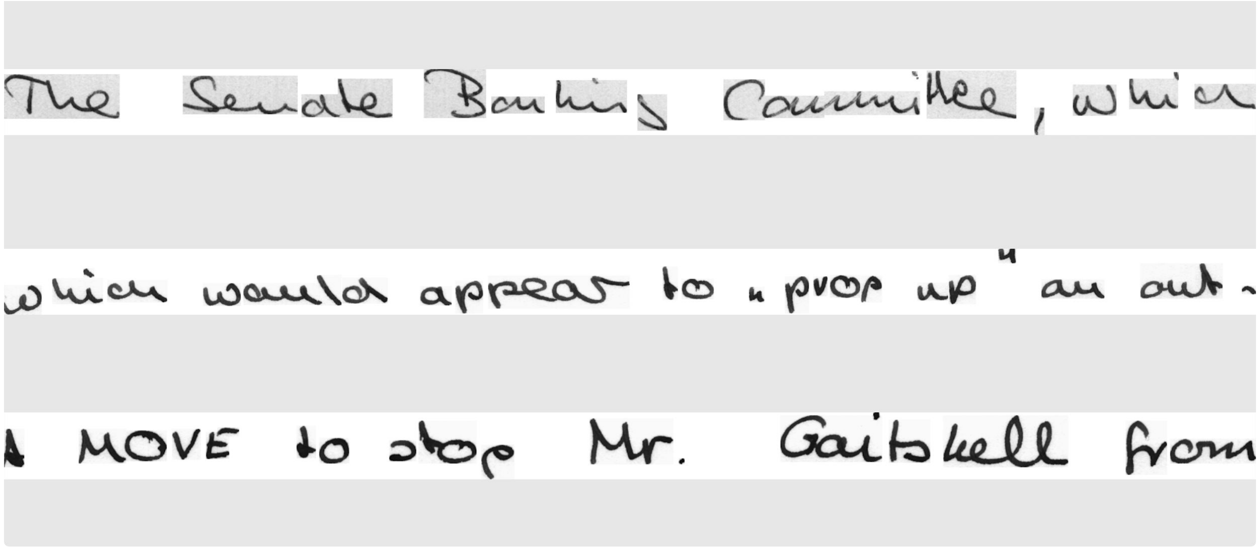
Forms_for_parsing.txt



```
a01-003 002 3
a01-003u 000 3
a01-003x 003 3
a01-007 004 3
a01-007u 000 3
a01-007x 003 3
a01-011 005 2
a01-011u 000 2
a01-011x 006 2
a01-014 007 5
```

Dataset_subset

In total: 4988 number of images

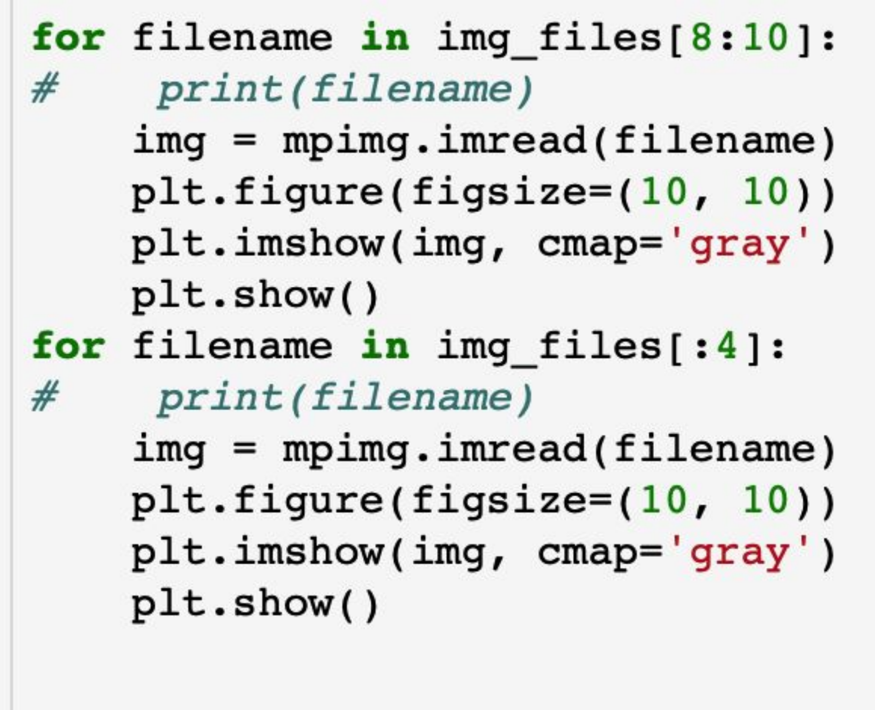


The Senate Banking Committee, whic

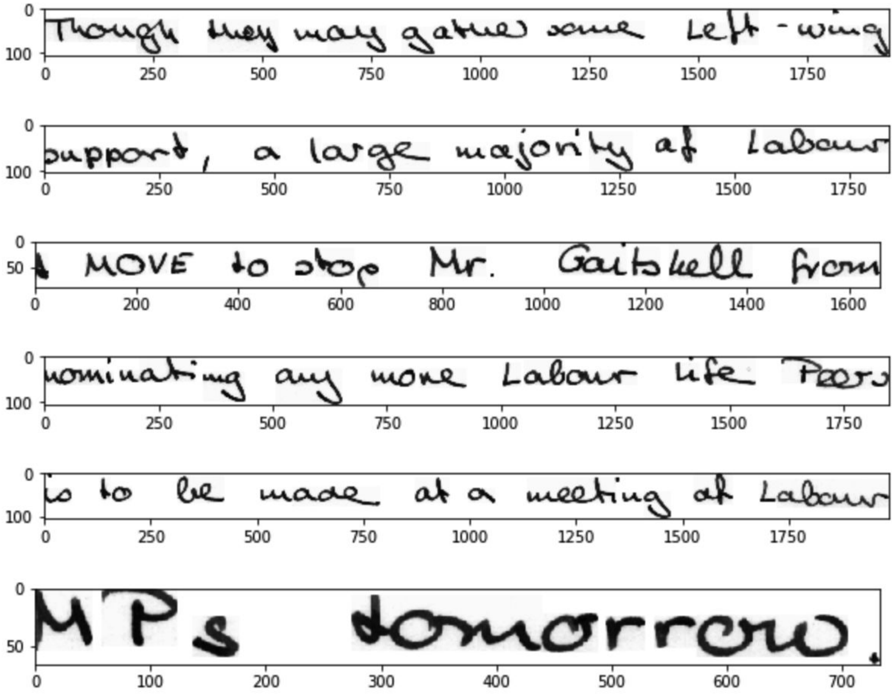
which would appear to "prop up" an out-

A MOVE to stop Mr. Gaitskell from

Handwritten scripts by one writer

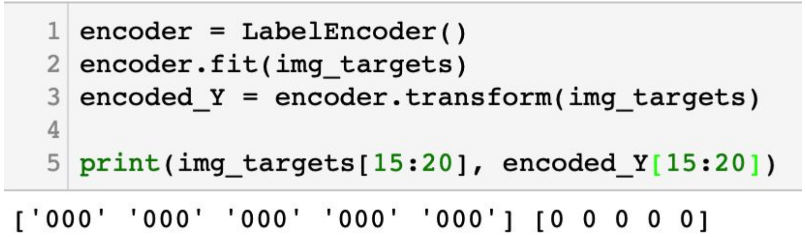


```
for filename in img_files[8:10]:
#     print(filename)
    img = mpimg.imread(filename)
    plt.figure(figsize=(10, 10))
    plt.imshow(img, cmap='gray')
    plt.show()
for filename in img_files[:4]:
#     print(filename)
    img = mpimg.imread(filename)
    plt.figure(figsize=(10, 10))
    plt.imshow(img, cmap='gray')
    plt.show()
```



Data Pre-Processing

1. Label Encode

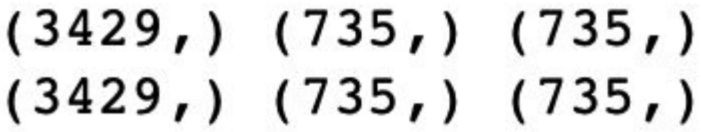


```
encoder = LabelEncoder()
encoder.fit(img_targets)
encoded_Y = encoder.transform(img_targets)

print(img_targets[15:20], encoded_Y[15:20])
```

```
['000' '000' '000' '000' '000'] [0 0 0 0 0]
```

2. Train, Validation, Test: 7:1.5:1.5

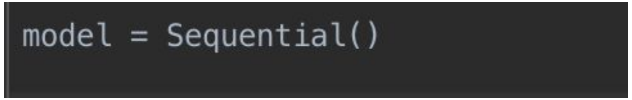


```
(3429,) (735,) (735,)
(3429,) (735,) (735,)
```

3. Resize

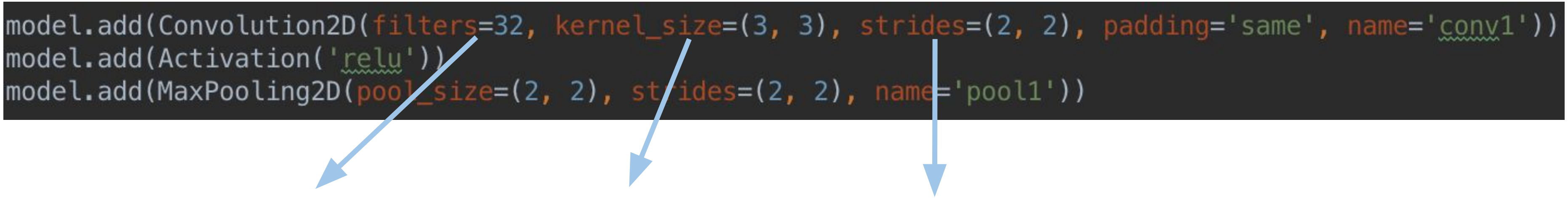
Network Construction

Declares the model type as Sequential().



```
model = Sequential()
```

Add a 2D convolutional layer to process input images



```
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'))
```

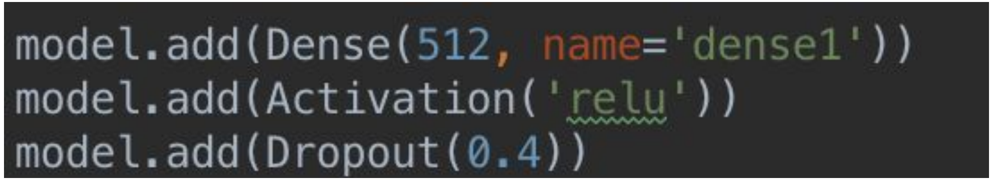
number of output channels

size of moving window

strides in the *x* and *y* directions

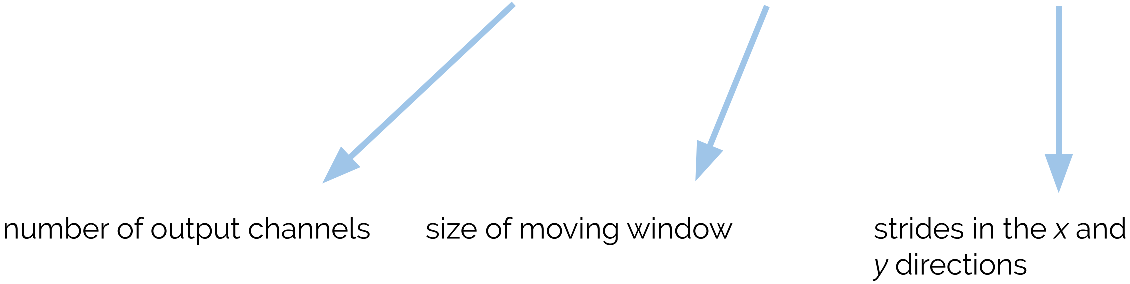
Network Construction

Network Construction



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

Flatten the output from these to enter our fully connected

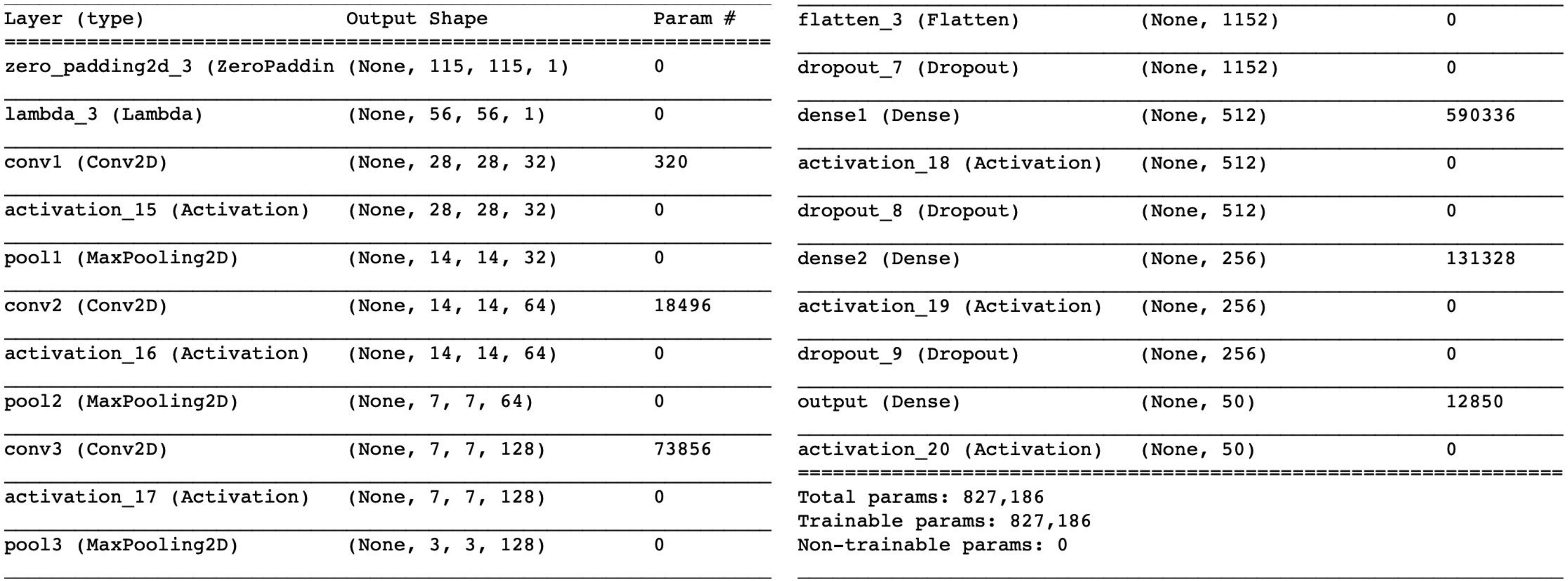


number of output channels

size of moving window

strides in the *x* and *y* directions

Architecture Network

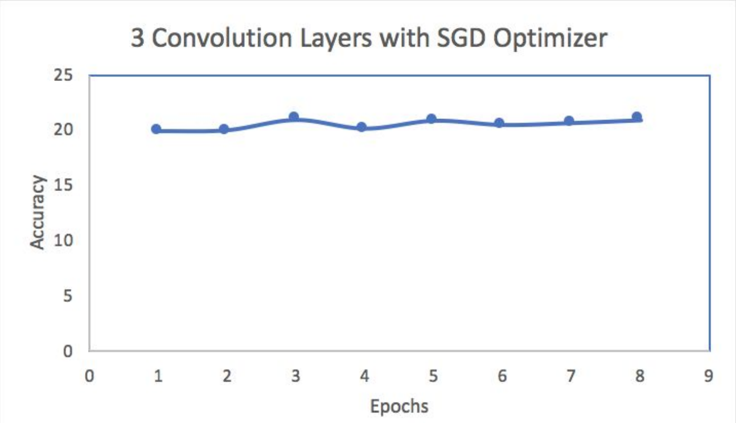


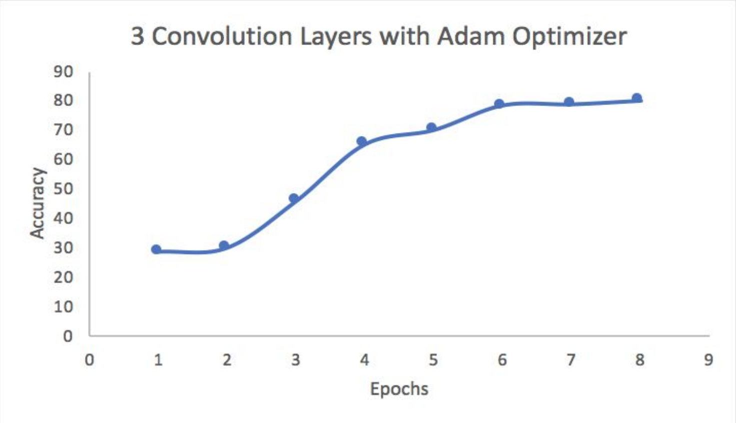
```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
zero_padding2d_3 (ZeroPaddin (None, 115, 115, 1)       0
_________________________________________________________________
lambda_3 (Lambda)            (None, 56, 56, 1)         0
_________________________________________________________________
conv1 (Conv2D)               (None, 28, 28, 32)        320
_________________________________________________________________
activation_15 (Activation)   (None, 28, 28, 32)        0
_________________________________________________________________
pool1 (MaxPooling2D)         (None, 14, 14, 32)        0
_________________________________________________________________
conv2 (Conv2D)               (None, 14, 14, 64)        18496
_________________________________________________________________
activation_16 (Activation)   (None, 14, 14, 64)        0
_________________________________________________________________
pool2 (MaxPooling2D)         (None, 7, 7, 64)          0
_________________________________________________________________
conv3 (Conv2D)               (None, 7, 7, 128)         73856
_________________________________________________________________
activation_17 (Activation)   (None, 7, 7, 128)         0
_________________________________________________________________
pool3 (MaxPooling2D)         (None, 3, 3, 128)         0
_________________________________________________________________
```

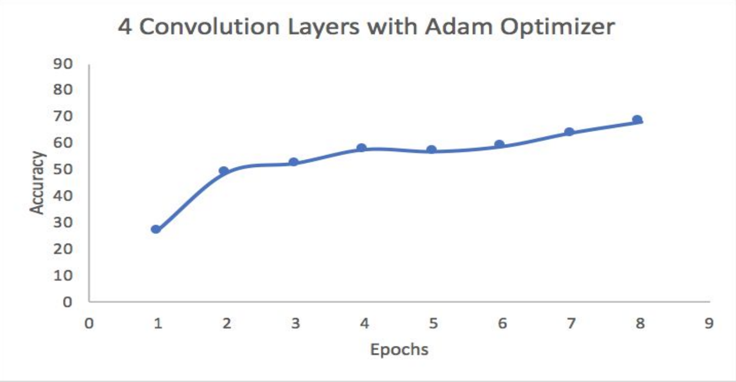
```
_________________________________________________________________
flatten_3 (Flatten)          (None, 1152)              0
_________________________________________________________________
dropout_7 (Dropout)          (None, 1152)              0
_________________________________________________________________
dense1 (Dense)               (None, 512)               590336
_________________________________________________________________
activation_18 (Activation)   (None, 512)               0
_________________________________________________________________
dropout_8 (Dropout)          (None, 512)               0
_________________________________________________________________
dense2 (Dense)               (None, 256)               131328
_________________________________________________________________
activation_19 (Activation)   (None, 256)               0
_________________________________________________________________
dropout_9 (Dropout)          (None, 256)               0
_________________________________________________________________
output (Dense)               (None, 50)                12850
_________________________________________________________________
activation_20 (Activation)   (None, 50)                0
=================================================================
Total params: 827,186
Trainable params: 827,186
Non-trainable params: 0
_________________________________________________________________
```

3-Layer CNN (Adam Optimizer) Output

Result


3 Convolution Layers with SGD Optimizer


3 Convolution Layers with Adam Optimizer


4 Convolution Layers with Adam Optimizer

Training Loss Plots

Summary and Conclusion

Keras

Adam & SGD

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/