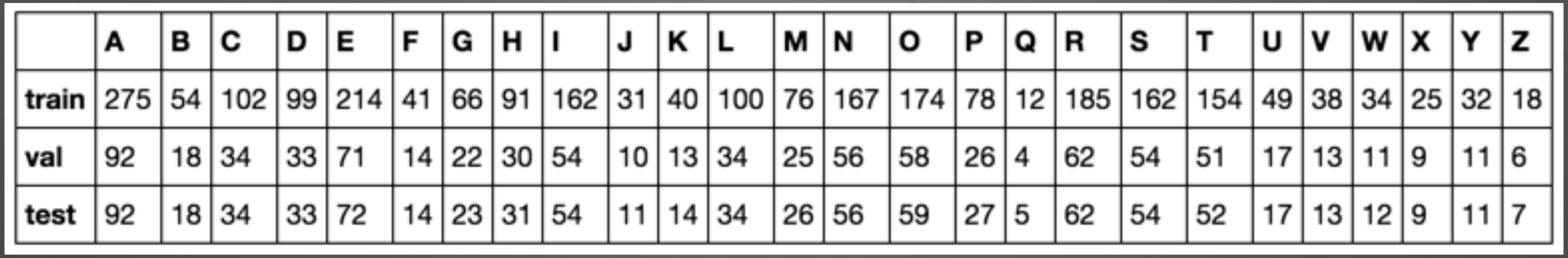
Character Recognition

Michael Seeber

Letter Frequency



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
train	275	54	102	99	214	41	66	91	162	31	40	100	76	167	174	78	12	185	162	154	49	38	34	25	32	18
val	92	18	34	33	71	14	22	30	54	10	13	34	25	56	58	26	4	62	54	51	17	13	11	9	11	6
test	92	18	34	33	72	14	23	31	54	11	14	34	26	56	59	27	5	62	54	52	17	13	12	9	11	7

- Stratified Sample: Training, Validation, Test

Modeling Process

- Completed 11 runs with reports
- Each run fits multiple models
- Run results inform the next run

Run Reports

- Reports Include:
  - Hyper-parameter information for the run
  - Epochs, Run Time, Training & Validation Accuracy
  - Model Architectures
  - Training Plots
  - Visualizations of Dense and Convolutional Layers
  - Accuracy by Letter
  - Sample Images for Correct and Incorrect Predictions

- Reports are linked in the subsequent slides

Run 1

- Question: Starting simple, using only dense layers, how well can we fit the data?

- Models: 4 models with different depths

- Best Training Accuracy: 0.93
- Best Validation Accuracy: 0.69

- Observations:
  - Need to increase capacity to get closer to 1 on training
  - Overfitting training relative to validation
  - Unable to identify “Q” in the validation data

https://github.com/mike-seeber/Character/blob/master/reports/run1/report.md

Run 2

- Question: Does adding convolutions improve results?

- Models: 4 simple architectures with convolutions of varying depth

- Best Training Accuracy: 0.98
- Best Validation Accuracy: 0.78

- Observations:
  - Improved capacity
  - Improved validation accuracy
  - Still overfitting training versus validation
  - Unable to identify “Q” in the validation data

https://github.com/mike-seeber/Character/blob/master/reports/run2/report.md

Run 3

- Question: Does adding data augmentation improve results?
- Models: Same 4 architectures from Run 2, now with augmentation
- Best Training Accuracy: 0.72
- Best Validation Accuracy: 0.81
- Observations:
  - Validation accuracy improved
  - Training data is under-fitting
  - Unable to identify “Q” in the validation data

https://github.com/mike-seeber/Character/blob/master/reports/run3/report.md

Run 4

- Question: Prior runs used early stopping on validation accuracy. Does a switch to training accuracy make sense now that we have incorporated data augmentation?
- Models: Same 4 architectures from Run 3
- Best Training Accuracy: 0.74
- Best Validation Accuracy: 0.83
- Observations:
  - Validation accuracy improved
  - Training data accuracy is higher, but still under-fitting
  - We correctly identified a validation “Q”!

https://github.com/mike-seeber/Character/blob/master/reports/run4/report.md

Run 5

- Question: Can we improve by increasing capacity?

- Models: Same 4 architectures from Run 3 with double the filters in each layer. 5th architecture with another set of convolutional layers and padding.

- Best Training Accuracy: 0.86
- Best Validation Accuracy: 0.86

- Observations:
  - Big improvement in training and validation accuracy!
  - New, deepest model achieved the best results

https://github.com/mike-seeber/Character/blob/master/reports/run5/report.md

Run 6

- Question: Can we improve by increasing capacity again?
- Models: Biggest architecture from Run 5 plus 4 new bigger, deeper architectures
- Best Training Accuracy: 0.91
- Best Validation Accuracy: 0.90
- Observations:
  - Result is better again. However, the best model this run is the same as the best model from Run 5 with just a different result

https://github.com/mike-seeber/Character/blob/master/reports/run6/report.md

Run 7

- Question: Are we stopping too early?

- Models: Same as Run 6 with early stopping min delta changed to .001

- Best Training Accuracy: 0.94
- Best Validation Accuracy: 0.89

- Observations:
  - Training took much longer
  - Training accuracy improved a little but validation did not
  - Did not accept changes from this run

https://github.com/mike-seeber/Character/blob/master/reports/run7/report.md

Ad Hoc

- Run 7 ran for a long time, so I tried a couple ad hoc tactics while it was running (no reports available):
  - I added another dense layer to the best model with no improvement
  - I adjusted the training data to equally represent the same number of samples from each letter and fed this data to the augmentation. The hypothesis was this might help by providing more augmentations of scarce letters. However, accuracy declined a little. I believe this decline may be a result of the model taking advantage of knowing the class imbalance to improve the model predictions.
  - I tried changing the size of the kernels in the convolutional filters from 2 to 3 and 4 with no improvement.

Run 8

- Question: We are using data augmentation as one method to prevent overfitting. Does introducing dropout improve results? Possibly by reducing complex co-adaptations among the neurons and benefiting from ensembling

- Models: Kept 4 of the largest models from Run 7. Dropped the largest which wasn't justifying the run time

- Best Training Accuracy: 0.88
- Best Validation Accuracy: 0.89

- Observations:
  - Results did not improve
  - Did not accept changes from this run

https://github.com/mike-seeber/Character/blob/master/reports/run8/report.md

Run 9

- Question: Can I find better augmentation parameters?

- Models: Kept best model. Ran 60 models, random sampling from data augmentation parameters on AWS GPU

- Best Training Accuracy: 0.93
- Best Validation Accuracy: 0.89

- Observations:
  - None of the sampled parameters improved the results over the defaults I previously selected
  - Increasing rotation seemed to have the largest negative effect on decreasing performance

- (Created different report format for Runs 9 and 10)

https://github.com/mike-seeber/Character/blob/master/reports/run9/report.md

Run 10

- Question: Can I find optimal number of convolutional filters and dense units?

- Models: Continued with best model. Ran 30 models, random sampling the filter and unit parameters.

- Best Training Accuracy: 0.94
- Best Validation Accuracy: 0.90

- Observations:
    - Running this many models will have random fluctuations in results just by chance. It appears that it is important to have the highest number of filters in the 3rd convolutional layer, but okay to have a smaller number of filters in prior layers
    - Does not seem sensitive to number of units in dense layer
    - Previous best architecture already adheres to the findings, and no changes kept from this run

https://github.com/mike-seeber/Character/blob/master/reports/run10/report.md

Run 11

- Question: Do any other Keras Optimizers improve results?

- Models: Continued with best model. Fit with 7 different optimizers.

- Best Training Accuracy: 0.90
- Best Validation Accuracy: 0.88

- Observations:
  - Reasonable results for all optimizers except SGD (with default parameters)
  - No change from using Adam

https://github.com/mike-seeber/Character/blob/master/reports/run11/report.md

Test Accuracy

- Refit the best model to the training data and computed training, validation, and test accuracy
- In this instance, I am reporting training accuracy on the true training data without data augmentation
- This is the first time any model has seen the test data

- Training Accuracy: 97.7%
- Validation Accuracy: 87.4%
- Test Accuracy: 89.5%

https://github.com/mike-seeber/Character/blob/master/reports/final_model_accuracy.md

Feature Learning

- How well did the convolutional layers of our best model learn features relevant to character recognition?
- Are the features good for a task like…

Bengali Digit Recognition

Feature Test

- Fit 4 models with the same dense layers

- Model 1: Training data is the Bengali Digit data
- Model 2: Training data is constructed by inputting the Bengali Digit data into our best character model and outputting the data at the end of the first set of convolutional layers (conv/conv/pool)
- Models 3-4: Same as Model 2, but training data obtained by outputting from the second and third set of convolutional layers, respectively

Feature Results

- Features are Bengali Digit Data
  - Test Accuracy: 86.3%
- Features from Character Model Pool #1
  - Test Accuracy: 94.7%
- Features from Character Model Pool #2
  - Test Accuracy: 96.8%
- Features from Character Model Pool #3
  - Test Accuracy: 94.0%
- The Features learned in our Character Recognition Model are Useful when applied to other character sets, even in another language!

https://github.com/mike-seeber/Character/blob/master/reports/bengali_accuracy.md

Appendix

character_data.py

- read_characters()
  - Loads each image file and converts to numeric and grey-scale
  - Loads image labels and converts to numeric (A=0, B=1, etc.)
  - Returns X and y
- letters(y_sample)
  - For convenience, converts from numeric back to letters (0=A, B=1, etc.)
- read_bengali()
  - Loads each image file and converts to numeric and grey-scale
  - Loads image labels
  - Returns X and y

https://github.com/mike-seeber/Character/blob/master/code/character_data.py

sample_split.py

- train_val_test_strat(X, y, split=(.6, .2, .2))
    - Stratified split between training, test, and validation
    - I use this function once and pickle the tensors to use the same split throughout all of modeling
- Xy_sample(X, y=None, size=5)
    - For convenience, take a random sample from X and y (y is optional)
- y_categorical(*y)
    - Send a list of y's (for example: y_train, y_val, y_test), and turn each y into a one-hot vector

https://github.com/mike-seeber/Character/blob/master/code/sample_split.py

visuals.py

- letter_subplots_train_test_split(letter, X_train, X_val, X_test, y_train_num, y_val_num, y_test_num)
  - For the provided letter, sample 5 images from train, val, and test

- make_subplots(X_sample, y_sample=None, columns=5, save=None, close=True)
  - For the given sample, display all of the images

- train_val_test_composition(y_train, y_val, y_test)
  - Output a dataframe visual with the number of training, validation, and test samples by letter

https://github.com/mike-seeber/Character/blob/master/code/visuals.py

unit_tests.py

- Assert statements to test the functions from:
    - character_data.py
    - sample_split.py
    - visuals.py

https://github.com/mike-seeber/Character/blob/master/code/unit_tests.py

model_step0_functions.py

- Functions primarily used to create the model run reports
- pickle_rw(*tuples, folder='../pickle/', write=True)
  - For convenience, pickle objects to and from file
- architecture(model)
  - Extract the model architecture and add to the report
- training(history, run_id, model_id)
  - Save the training/validation accuracy plot and add to the report
- weights(model, run_id, model_id)
  - Save the visualizations of the layer weights and add to the report
- ltr_accuracy(model, X_train, y_train_num, X_val, y_val_num)
  - Calculuate train/validation accuracy by letter and add to the report
- validation_plots(model, X_val, y_val_num, run_id, model_id)
  - Save random correct/incorrect letter images from validation set and add to the report
- report_save(start_time, report, model_id, model, history, run_id, results, X_train, y_train_num, X_val, y_val_num)
  - Update all the results for the report

https://github.com/mike-seeber/Character/blob/master/code/model_step0_functions.py

model_step1_data.py

- Read in the original image data from file
- Perform the stratified split and pickle the tensors

https://github.com/mike-seeber/Character/blob/master/code/model_step1_data.py

model_step2_run#.py

- Define the deep learning models for the run (run #’s from 1 to 11)
- Fit each model in the run and create the run report

https://github.com/mike-seeber/Character/blob/master/code/model_step2_run1.py

model_step3_test.py

- Fit the best model and produce the accuracy on training, validation, and test
- Save Final Model Accuracy report

https://github.com/mike-seeber/Character/blob/master/code/model_step3_test.py

bengali.py

- Fit 4 models with dense layers to the Bengali Digit data
- 3 of the models contain frozen convolutional and maxpool layers from the best Character model
- Create and save Bengali Accuracy report

https://github.com/mike-seeber/Character/blob/master/code/bengali.py