Jithin Joyson December 8, 2018

CS 584 Report

Final Results (Best F1)

Phases	Classifier	Data Set	F1 Score	Parameters
1	KNeighbors	Bush	0.1416768033	N_neighbors = 1

1	SVC	Bush	0.6404965167	Kernel = poly
				Degree = 1
				Gamma = 0.1

2	KNeighbors (PCA)	Bush	0.173493245	N_neighbors = 1
				N_components = 48
				Whiten = False
				Svd_solver = full

2	SVC (PCA)	Bush	0.6484317038	Kernel = linear
				C = 0.1
				N_components = 2181
				Whiten = False
				Svd_solver = randomized

3	CNNs	Bush	0.9271137026239066	Diagram
4	Transferred (CNNs)	Bush	0.9053000000000001	Diagram
1	KNeighbors	Williams	0.03508772	N_neighbors = 1

1	SVC	Williams	0.5239316233	Kernel = linear
				C = 100

2	KNeighbors	Williams	0.2452475685	N_neighbors = 1
				N_components = 13
				Whiten = False
				Svd_solver = auto

2	SVC	Williams	0.5239316239	Kernel = linear
				C = 0.1
				N_components = 4096
				Whiten = False
				Svd_solver = auto

3	CNNs	Williams	0.7142857142857143	Diagram
4	Transferred (CNNs)	Williams	0.6923076923076924	Diagram

Metric

Throughout this project, the f1 score was the most important metric used to evaluate models. This metric is 2*precision*recall/ (precession + recall). Thus, this metric is the harmonic mean (or balance) between recall and precision. Maximizing this metric yields a better model when the target class in the dataset is rare (as is for our case).

Phase 1

Testing Parameters:	KNN - {n_neighbors: [1,3,5]}
	SVC – {kernel: ["linear", "poly", "rbf"], degree: [.01, .1, 1, 10, 100], gamma: [.01, .1, 1, 10, 100], C: [1, 10, 100]}

The KNeighbors classifier was run 3 times on both data sets for 1, 3, and neighbors. Testing this classifier was simple since the total number of iterations were about 6. As the number of neighbors increased, the computation time also increased. Due the lack of computation speed on my personal computer, the fusion cluster was used. The best parameter for this classifier was n_neighbors = 1 since the dataset had lot more skewed a lot of negatives. Increasing the number of neighbors to vote on how to classify a new data would increase the likelihood of a lot of false negative classifications. The SVC classifier was run 225 times on both data sets for a combination of kernels (3), degree(5), C(3) and gamma (5). Testing this classifier was more complex since there was free will on what gamma and degree values where chosen to test the model. As the degree and gamma values increased, the computation times increased. Once again (and for the rest of the computations), the fusion cluster was used since this can be run when I am not active (sleeping) or when I have work to do. It was run a total of 150 times for both models. The best result was poly since gamma, C, and degree added complexity to a linear kernel and can be customized for the data set.

Phase 2

Testing Parameters:	PCA – {svd_solver: ["auto", "full"], whiten: [True, False], n_components: [1-81, 381, 681, 981, 1281, 1581, 1881, 2181, 2481, 2781, 3081, 3381, 3681, 3981, 4096]}

KNN - {n_neighbors: [1,3,5]}

SVC – {kernel: [“linear”, “poly”, “rbf”], degree: [1, 10], gamma: [1, 10], C: [.01]}

Both KNeighbors and SVC testing was similarly as phase 1 testing except with additions of iterations brought on by PCA. Thus, there was lot more iterations on the datasets. The biggest problems was kernel = “rbf” which refuse to run after n_components > 1000. Therefore, there was less iterations done on this parameter but was quite a lot. It was interesting to see variability in the n_components parameter in the models; other than this parameter, every other is similar throughout both final classifiers.

Phase 3

Diagram attached to end since it would not fit in portrait mode.

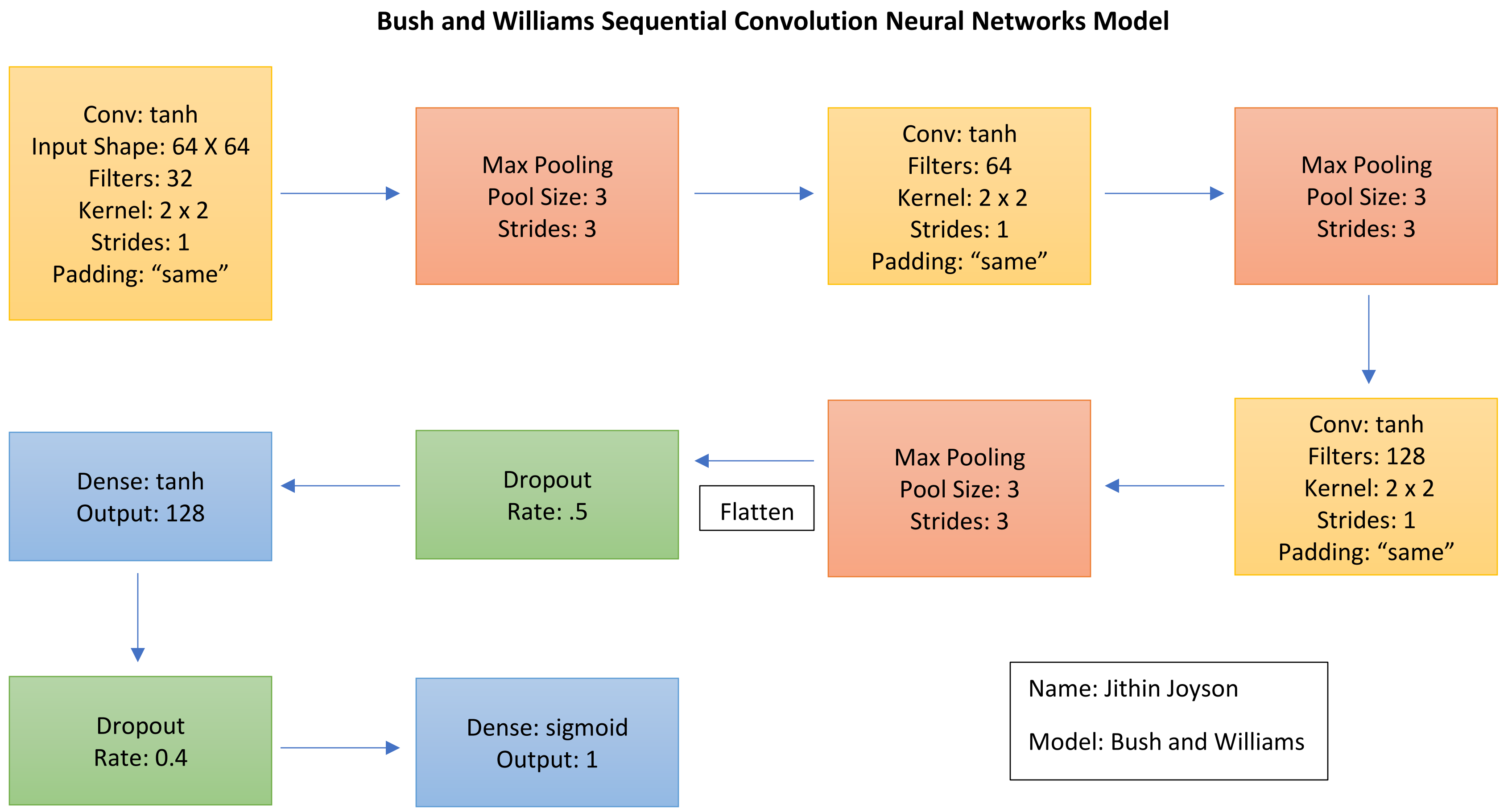
The diagram specifies different parameters tried for this model. Since it is hard to evaluate different activation functions when layered upon each other, this phase purely a test of going through all the iterations not knowing if the final f1 score would get better or worse after another one. It was interesting to see a higher batch size and lower drop in dropout resulting in better models. Combining activation functions did not yield a better model but keep all the functions the same was the answer

Phase 4

URL link - https://www.kaggle.com/shuchirb/olivetti-faces

# of images – 400

The chosen dataset consisted of 400 images (10 0f 40 people). One of the people were chosen as the target (1) and any other image (390) was classified as 0. When trying to find the best f1 score for this model, a similar test of Phase 3 was done. It yielded in .8 for this dataset with the same parameters from phase 3 (same diagram). Then the model was used to fit both bush and Williams datasets. Interestingly enough, the f1 score decreased down. This makes sense since even though we added complexity to the model, it falsely classified our bush and Williams models (since the older dataset had bearing in the decisions.



Several Loops were run trying to find a model that outputted a f1 score of 1.0 on the train data and the highest f1 score on the test data. Parameters that were tested are listed out in each layer or step. Ranges of values included a combination of the following:

Activation Function: {tanh, relu, softmax, sigmoid}	Note: Combination of these for the 4 different instances (3 CONV and 1 Dense)
Kernel: {3, 4, 5}	
Filters: {32, 64, 128, 256}	Note: The next filter would be larger than the previous (3 CONV)
Pool Size: {3, 4, 5}	
Dropout Rate: {.1,.2,.3,.4,.5,.6,.7,.8,.9}	Note: This step was added to combat overfitting the data
Loss: {binary_crossentropy, poisson}	Note: This was a parameter in the compile step and **binary_crossentropy** was the best

The Input Shape was (64,64,**1**) where **1** specified that the images were grayscale.
Both the Bush and Williams models happened to have the largest test f1 score with the same model as above when the train f1 score was the largest (1.0).