Authors: Ze Hong Wu, Ying Jie Mei

Introduction

This report describes the work we carried out during Stage IV.

This report will be structured to include the following sections:
- Part 1: Changes attempted by each team member; Results and Observations
- Part 2: Model Architecture
- Part 3: Proposed Next Steps

Part 1: Changes Tested

At the end of Stage III, we selected a model based on the DenseNet architecture. The architecture was described in Part 3 of the Stage III report and will be revisited again in Part 3 of this report.

During Stage IV of this project, we tested the following changes to the DenseNet model:

Ze Hong Wu: Gradient Clipping, Activation functions, compression factor, learning rate, growth rate
Ying Jie Mei: Optimizer, Regularization, Dropout, Early Stopping

We will discuss the meaning and significance of compression factor and growth rate during the corresponding subsections.

Part 1.1: Work by Ze Hong Wu

I started my work by using the model submitted for Stage III as a baseline. During this process, I corrected an error in my TFDataset files that caused the labels to be improperly generated, resulting in large discrepancies between model performance during training and when tested against the holdout set. I re-trained the Stage III model on this fixed dataset, without any modifications to the architecture or hyperparameters. The baseline Loss - Accuracy - F1 Macro score is presented below.

Table 1: Loss - Accuracy - F1 Macro values for the baseline comparison model.

Current model	Based on	Loss	Accuracy	F1 Macro
Baseline	None	0.6446	0.7686	0.7696

Part 1.1.1: Changes Tested

I tried out the following changes to the model, explained in more detail:

- Initial Corrections: I removed an incorrectly added transition block that was not supposed to be in the architecture, fixed a bug that unintentionally set the initial learning rate to 0.01 (intended value is 0.1), and increased training epoch count to 24 epochs. This change is applied to the baseline.
- Gradient Clipping: I applied “clipnorm=1.0” to the SGD optimizer used to optimize the model. This change uses the Initial Corrections model as a baseline.
- Leaky ReLU: I changed the activation function of all convolutional layers in the model from ReLU to Leaky ReLU. This change uses the Gradient Clipping model as a baseline.
- Compression Factor: I changed the compression factor for the transition blocks in the model from 0.5 to 0.66. This change uses the Gradient Clipping model as a baseline due to poor training performance on the Leaky ReLU model.
    - Compression Factor (CF) is a number that determines how many filters are preserved when going through a transition block. For example, with a CF of 0.75, the number of filters after passing through the transition block will change from X to 0.75X.
- Growth Rate: I changed the growth rate of the residual blocks from 32 to 48. This change uses the Compression Factor model as a baseline.
    - Growth Rate (GR) is a number that determines the rate at which the number of filters change after each residual block. A residual block with a GR of 32 might take in an input of 128 filters, pass it through its convolution layers to produce an output of 32 filters, then concatenate the two to produce a true output of (32+128)=160 filters.
- Learning Rate: I changed the learning rate (lr) scheduler from a 90% reduction to a 50% reduction. This change uses the Compression Factor model as a baseline due to poor results from the Growth Rate model.
    - The DenseNet paper favored a lr scheme with an initial lr of 0.1 and a multiplication by 0.1 (90% reduction) when the epochs elapsed reached the 50% and 75% mark. I changed the multiplier from 0.1 to 0.5 to achieve a 50% reduction instead.

Part 1.1.2: Experimental Results

The results of these changes are shown below.

Table 2: Full table of changes tested and experimental results. Loss, Accuracy, and F1 Macro scores are derived from evaluating the models against the test set.

Model	Based On	Changes	Loss	Accuracy	F1 Macro
Baseline	none	Baseline from Stage III	0.6446	0.7686	0.7696
Initial	Baseline	Removed transition layer	0.5793	0.8049	0.8041

		Fixed learning rate bug Trained for 24 epochs			
ClipNorm	Initial	Added clipnorm=1.0 to SGD	0.5196	0.8428	0.8404
LeakyReLU	ClipNorm	Changed ReLU for Leaky ReLU	0.4755	0.8263	0.8245
Compression	ClipNorm	Compression factor changed from 0.5 to 0.66	0.4658	0.8433	0.8422
Growth Rate	Compression	Growth Rate changed from 32 to 48	0.5159	0.8277	0.8252
Learning Rate	Compression	Learning Rate changed from -90% at 12 and 18 epochs to -50% at 12 and 18 epochs	0.5168	0.8134	0.8141

During each test I chose the previous model to base my testing work on using the following decision tree:

- I have a baseline model. I replicate its architecture and hyperparameters, changing only one feature, and create and train a new model Model1.
- If Model1 produces greater accuracy compared to baseline during evaluation, I will use Model2 as a baseline for my next feature test.
- If Model1 produces lesser accuracy, I will use the original baseline.
- Repeat steps until all features are tested.

I abandoned the Growth Rate and LeakyReLU changes instead of iterating upon them as a result of their reduced accuracy compared to the models they are based on.

Part 1.2: Work by Ying Jie Mei

The model we submitted in stage III gave me the following results.

Table 3: Loss, Accuracy, and F1 Macro score for the baseline model.

Baseline Model	Loss	Accuracy	F1_Score
my_densenet	0.5614	0.7959	0.7962

Starting from the baseline model, I made some modifications and tried out different approaches to get the best model result on my end so that later on we could combine our models' modifications that did better on accuracy and f1_score to be implemented towards the final model.

Table 4: Full table of changes tested and experimental results.

Initial Model	Model Name	Modification	Value	Loss	Accuracy	F1_Score
Baseline	M1	Optimizer	Adam	0.3813	0.8574	0.8589
M1	M2	Regularization	Dropout layer after activation function	0.9873	0.5918	0.5835
M1	M3	Filter	Increment base filter after each dense block	0.4921	0.8125	0.8157
M1	M4	Regularization	Dropout layer after convolution layers	0.8932	0.6523	0.6515
M1	M5	Image shape	Changed image shape to (256,256,3)	0.4403	0.8216	0.8243
M1	M6	Regularization Filter	Dropout layer after activation, before and after convolution, before and after concatenate, before and after transition layers, also filter scaling by 16 at end of each dense block	1.6273	0.3818	0.3372

The table above shows the different modifications I attempted from the baseline model. If the modified model's accuracy and f1_score are below the previous model, I will return the model to its previous state and continue from there. Some models' epochs remained unchanged throughout the training, the tests came back with early stopping results.

- Insert dropout layers after the activation function which is also before a convolution layer
- A dropout layer after convolution layers
- Dropout layer after concatenation
- Add dropout layers before and after the transition layers
- Changed image shape for input
- Incrementing filter count after each dense block
- Changed optimizer from SGD to Adam

For the changes in using dropout layers and modifying filter counts, the results came back as early stopping at a certain epoch, which means the model isn't learning anything after that epoch. Not only that but the accuracy and F1 score seem to be very low, so I chose to stop using dropout layers and modifying filter counts. As for the image shape, I modified a lower input and the results are similar but a bit lower than the baseline model, so I chose to drop the image shape modification as well.

Part 2: Model Architecture

The architecture of the model we will submit for Stage IV is largely identical to the architecture of our model for Stage III. Since our Stage IV final product focuses more on hyperparameters and other features that do not show up well on an architecture diagram, this part of the report will focus more on these less visible changes.

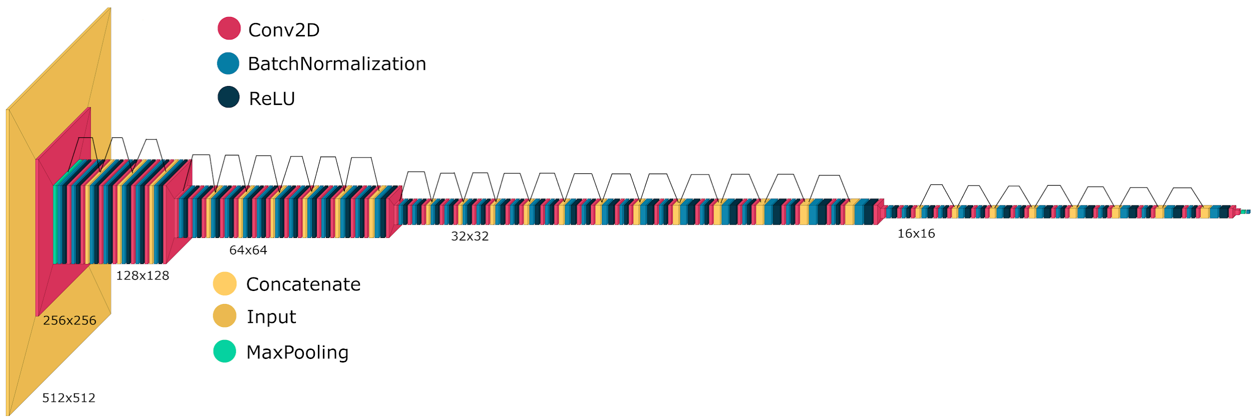

Figure X: Architecture of the chosen model, generated using the visualkeras package for Python and edited using GIMP and Paint. Black lines connecting different layers are skip connections. A full-resolution version of this image can be found here.

We chose to proceed with this model, which is the “M1” model from Table 4, after discussing our observations. While Ze Hong’s models contain a number of changes from the original, Ying Jie’s M1 model with the Adam optimizer outperformed all of them when judging by accuracy.

Normally we should combine our work by training a model using chosen changes from both team members, however as a result of improper time management we did not have time to do this. We will discuss our plans for remedying this in Part 3.

The chosen model is optimized using Adam with these hyperparameters:

- Learning rate: 0.001
- ClipNorm: 1.0

Additionally, the chosen model itself uses these hyperparameters:

- Growth Rate: 32 filters
- Compression Factor: 0.5
- Activation function: ReLU
- Metrics: Accuracy, F1 Macro
- Loss: Categorical Cross-entropy

- Epochs trained: 16
- Initial Learning Rate (lr): 0.1
- Learning Rate schedule: Reduce lr by 90% at 8 and 12 epochs

The architecture of this model is described in the following tables, taken from the Stage III report and updated to reflect our changes.

Table 3: Residual Block architecture

Layers	Parameters and Notes	Output shape
Input	Residual Block begins here	x,y,filters
BatchNormalization		x,y,filters
ReLU		x,y,filters
Conv2D	128 filters, 1x1 kernels, stride 1, same padding	x,y,128
BatchNormalization		x,y,128
ReLU		x,y,128
Conv2D	32 filters, 3x3 kernels, stride 1, same padding, output=b	x,y,32
Concatenate	Skip connection, links input with output	x,y,filters+32

Residual blocks are one of the complex components of this model. They consist of a pair of BatchNorm-ReLU-Conv2D stacks, one after the other, with a skip connection that concatenates the input with the output.

Table 4: Transition Block architecture

Layers	Parameters and Notes	Output shape
Input	Transition layer begins here	x,y,filters
BatchNormalization		x,y,filters
ReLU		x,y,filters
Conv2D	filters/2 filters, 3x3 kernel, stride 1, same padding	x,y,filters*0.5
AveragePooling2D	2x2 pooling, stride 2, same padding	(x,y,filters)*0.5

Transition Blocks are another complex component. They consist of a BatchNormalization layer and a ReLU activation followed by a Conv2D layer and an AveragePooling2D layer. The Conv2D and AveragePooling2D layers serve to down-size the filters by an amount proportional to the Compression Factor.

Table 5: DenseNet model architecture

Layers	Parameters and Notes	Output shape
Input		512,512,3
Conv2D	64 filters, 7x7 kernel, stride 2, same padding, relu	256,256,64
MaxPooingl2D	3x3 pooling, stride 2, same padding	128,128,64
Residual Block		128,128,96
Residual Block		128,128,128
Residual Block		128,128,160
Transition Layer		64,64,80
Residual Block x6	Filter counts after each block are: 112, 144, 176, 208, 240, 272. (x,y) dims remain at (64,64).	64,64,272
Transition Layer		32,32,136
Residual block x12	Filter counts after each block are: 168, 200, 232, 264, 296, 328, 360, 392, 424, 456, 488, 520. (x,y) dims remain at (32,32).	32,32,520
Transition Layer		16,16,260
Residual Block x8	Filter counts after each block are:290, 324, 356, 388, 420, 458, 484, 516. (x,y) dims remain at (16,16).	16,16,516
Transition Layer	Unintentionally added to the model during Stage III. It was removed in Ze Hong's tests but not in Ying Jie's tests. It will be removed in Stage V.	8,8,258
GlobalAveragePooling2D		None,258

Part 3: Proposed Stage V Plans

For the Stage V work, we propose to carry out the following tasks:
1: We will train a new model, combining Ying Jie's Adam optimizer changes and Ze Hong's several tested changes with positive effects on accuracy, to serve as a baseline for one last set of fine-tuning.

2: For this fine-tuning, we will focus on adjusting the learning rate and the compression factor. Ying Jie will test compression factors of 0.75, 1.0, and any other values we may consider during training and their effects on model accuracy. Ze Hong will test new learning rate schedules, described in detail in the table below.
3: Upon comparing results one last time, we will choose one team member to train the final model iteration on the full train+test+valid dataset on an undecided number of epochs (tentatively 48).

Table 5: Proposed fine-tuning variants for the learning rate schedule

Schedule	Initial learning rate	Learning rate changes
Baseline	0.001	None
Version 1	0.01	-90% at 12 and 18 epochs
Version 2	0.002	-50% at 12 and 18 epochs
Version 3	0.002	-0.0005 at 6, 12, and 18 epochs