Customizing VGG16 And MobileNet Architectures for Binary Classification Task

Data Science and Economics

The Module Statistical Methods for Machine Learning

Mohammadhossein Jafari(964548)

March 4, 2023

I declare that this material, which I now submit for assessment, is entirely my own work and has not been taken from the work of others, save and to the extent that such work has been cited and acknowledged within the text of my work. I understand that plagiarism, collusion, and copying are grave and serious offenses in the university, and accept the penalties that would be imposed should I engage in plagiarism, collusion, or copying. This assignment, or any part of it, has not been previously submitted by me or any other person for assessment on this or any other course of study.

Abstract

In this research, two different Convolutional Neural Network architectures, which are VGG16 and MobileNet, are customized for this specific binary classification task. For each architecture, there are four variants. The variants of MobileNet architecture perform better than the variants of VGG16. The chosen input image size is (64,64) instead of the standard size of (224,224), and both models have acceptable results for this unconventional input image size. The research shows adding additional layers to the pre-trained architectures such MobileNet can increase their accuracy to above 93 percent and decrease the zero-one loss to less than 7 percent.

1 Dataset And Data Preprocessing

This binary classification task includes 25000 images of cats and dogs. The dataset is balanced meaning that there are 12500 images for each category. The images are in the format of JPG, and they are colored.

It is obvious that in a deep learning task, the quality of input data has a vital influence on the performance of the output model. The more the data is accurate, the more the generated model is capable. To check the quality of the images in the dataset, a python script named "damage-image-detector" is written to detect damaged images. The script can be found in the GitHub repository. The script was run, and it detected damaged images which are image number 666 in the dog folder and image number 11702 in the cat folder.

The damage image detector works like this:

- It goes into the directory of each category and opens each image.
- If the IOError or the SyntaxError happens, the name of that image is shown.

The damaged images were deleted manually, and the new dataset is named "pure-cats-dogs".

2 Architectures

Convolutional Neural Networks (CNN) are an essential part of deep learning scope. CNN models have important applications in different tasks. They can be used for face detection, biometric checking, helping automation processes, and many other applications. There are a wide variety of CNN architectures. In this research two CNN architectures named VGG16 and MobileNet are used to do the task of binary classification. Both CNN architectures have been developed for the multiclass classification of 1000 objects. They are general architectures that are customized for this specific binary classification task.

2.1 VGG16

VGG16 architecture consists of 16 layers[2]. These 16 layers are divided into two parts. The first part includes 13 convolutional layers which are (3*3). The pooling layers are in the size (2*2) and the stride size is 2 for them. The remaining three layers are fully connected. Figure 1 shows the VGG16 Architecture.

2.2 MobileNet

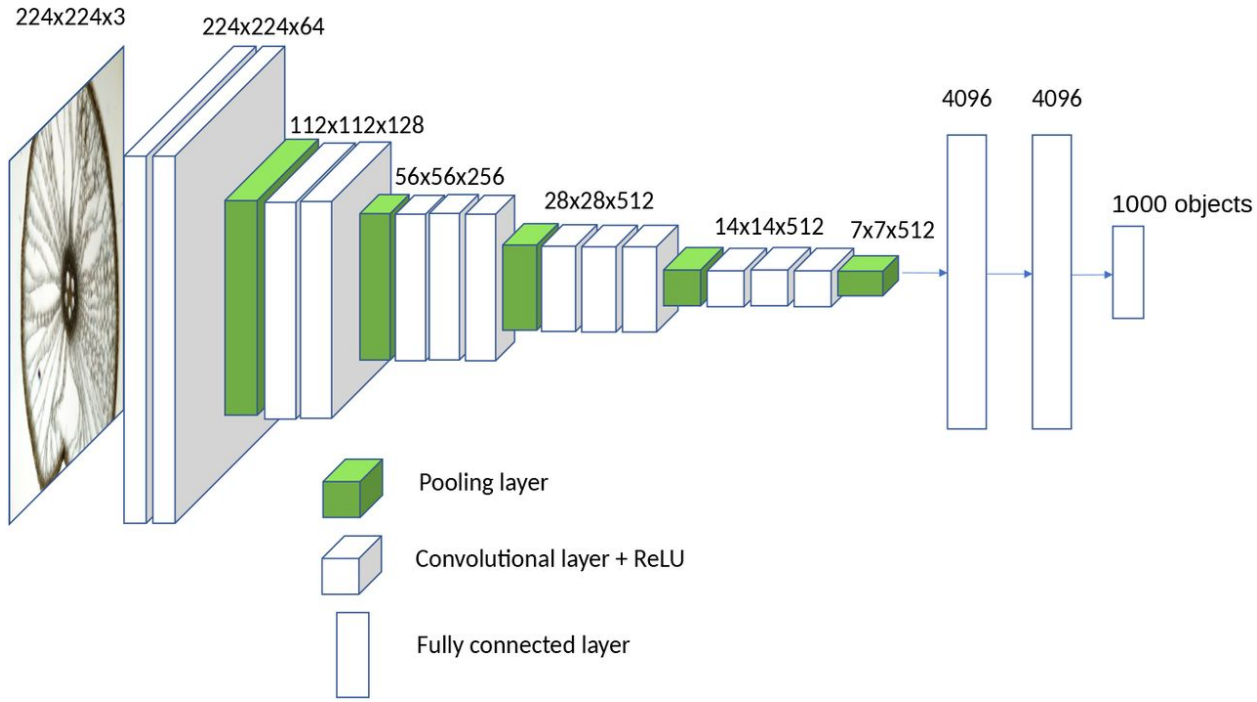
MobileNet has 13 depth-wise convolutional layers with the size (3*3), 1 convolution layer with the size (3*3), and 13 convolution layers with the size (1*1)[1]. Figure 2 shows the MobileNet Architecture.

3 Implementing the Models

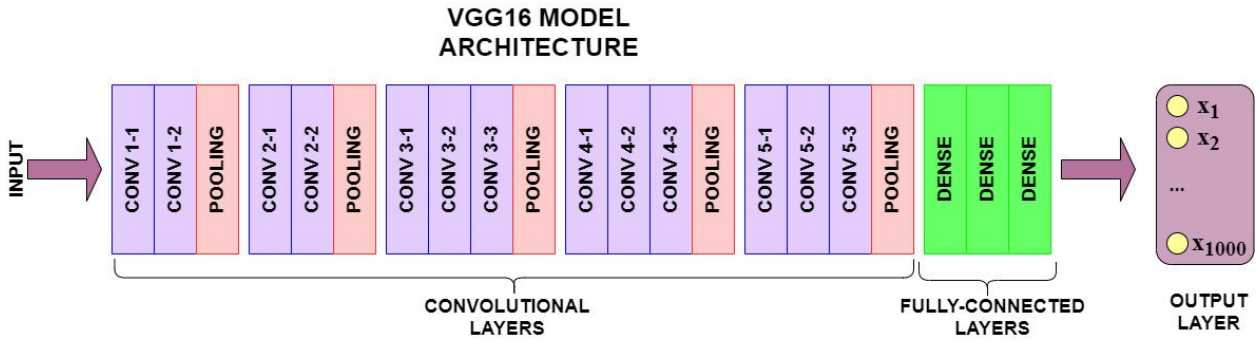
The VGG16 and MobileNet architectures are used to make 8 Models. For both VGG16 and MobileNet, there are 4 variants. When the models are trained, the loss is Binary Classification. After that, the 5-fold Cross Validation is run on the same architecture. For the cross-validation task, the loss is zero-one. Each model has two python files in the GitHub repository, one for training the model and the other one for doing the cross-validation. The batch size for all models is 32.

The inputs of both VGG16 and MobileNet architecture are RGB images, which have 3 channels. For both architectures, the standard size for the input images is (224,224). One of the goals of this work is to analyze the performance of both architectures in a different size which is pretty smaller. There were different options such as images in the sizes (32,32), (64,64), and (128,128). Finally, the size (64,64) was chosen. Therefore, the input of the architecture is (64,64,3).

ImgeDataGenerator allows the generation of batches of tensor images with real-time data augmentation. When the models are trained, an object named train-datagen is created with the help of ImageDataGenerator. It rescales images into the range of 0 to 1. The rescaling is useful to have better performance.



(a) Vgg16



(b) VGG16 Layers

Figure 1: Basic VGG16 Architecture

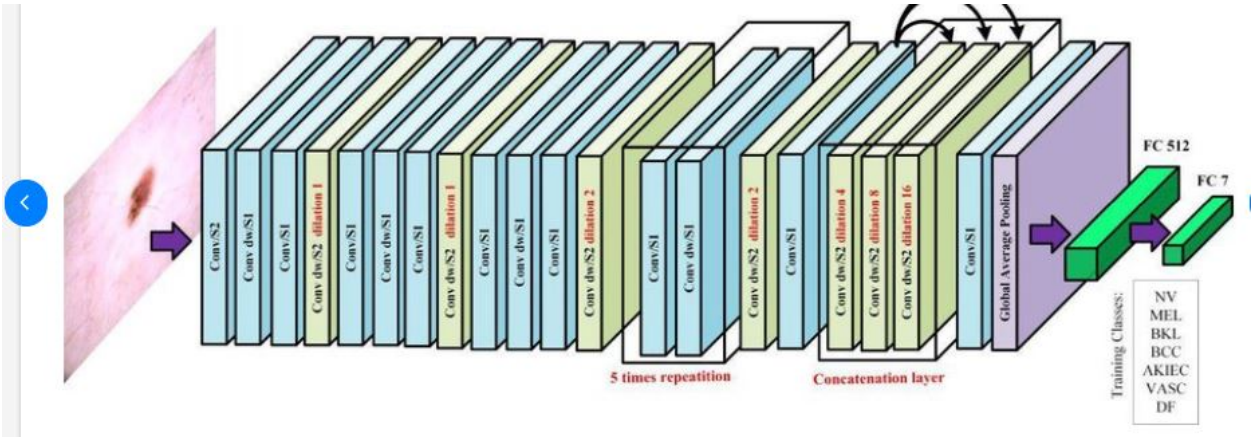


Figure 2: The MobileNet Architecture

3.1 Model 1

It has a VGG16 architecture as its basement. The output of the VGG16 basement is in the shape of (None, 2, 2, 512). A GlobalMaxPooling2D layer is added to change the output from (None, 2, 2, 512) to (None, 512). In the end, a Dense layer is added to make ready the output for the binary classification task. The optimizer for this model is Stochastic Gradient Descent. The learning rate of the optimizer is 0.0001 and its momentum is 0.9.

3.2 Model 2

It has the same architecture as Model 1 with the difference that the optimizer is Adam instead of the Stochastic Gradient Descent. The learning rate of the Adam optimizer is 0.001. It will be used to make a comparison between the performance of Stochastic Gradient Descent and Adam in this specific Binary classification task.

3.3 Model 3

One of the main purposes of this research is to add new layers to VGG16 and MobileNet architectures and evaluate the performance of these modifications. Model 3 starts with a VGG16 architecture. Then, there is a GlobalMaxPooling2D layer, a Dense layer with 512 units, and a Dropout layer with a rate of 0.5. The optimizer in this model is Stochastic Gradient Descent with a learning rate of 0.0001.

3.4 Model 4

It begins with a VGG16 architecture. Then, there is a GlobalMaxPooling2D layer, a Dense layer with 512 units, and a Dropout layer with a rate of 0.5. It has the same architecture as Model 3 with the difference that the optimizer is Adam with a learning rate of 0.001. It is the last model whose base architecture is VGG16. These additional layers add 262656 trainable parameters to architecture and the total number of parameters reached to 14,977,857 for this model.

3.5 Model 5

The MobileNet architecture is the skeleton of this model. A GlobalMaxPooling2D layer and a Dense layer are added to customize the model for the binary classification task. Its optimizer is Stochastic Gradient Descent with a learning rate of 0.0001 and momentum of 0.9.

3.6 Model 6

It has a similar architecture to Model 5 with the difference that its optimizer is Adam. The output of the MobileNet architecture is in the shape of (None, 2, 2, 1024), and it includes 3,228,864 parameters.

3.7 Model 7

A block which is consisted of a Dense layer with 512 units and a Dropout layer with a rate of 0.5 is added to model 5, and its location is between the GlobalMaxPooling2D layer and the final Dense layer. Its optimizer is Stochastic Gradient Descent with the same learning rate as other models whose optimizer is SGD.

3.8 Model 8

It has the same architecture as the Model 7. The additional block of layers adds 524800 trainable parameters to the basement architectures. Its optimizer is Adam.

4 Implementing the 5-Fold Cross-Validation

Considering the structure of the dataset, the images of each category are in a different directory and the method flow.from.directory of ImageDataGenerator is not compatible with the k-fold cross-validation method. Therefore, it is not possible to generate batches of tensor image data by ImageDataGenerator for doing the cross-validation.

The data which is needed for cross-validation is made ready from another approach. A python script named pickle-generator is written. The script can be found in the GitHub repository of the project.

The script defines for each class of objects a function that reads the images of that class and resizes them to (64, 64). A number 0 or 1 is allocated to the tensors of each image. This number is the label of the image based on whether the image is for a cat or a dog. The image and its label are stored in a list named "dataset".

The random.shuffle method is used on the "dataset" list to have more robust inputs. After that, two lists from the "dataset" are generated, one list is for the x-data and the other is for the y-data. The list of x-data is changed to a NumPy array. In the end, the x-data and y-data are stored separately, in the pickle format.

A pickle file is a byte stream generated by a python module named pickle. The advantage of this method is that it gives the ability to serialize pretty much any python object. The pickle files of images are used as the input of the cross-validation models. Cross-Validation Implementation: For each model, a different cross-validation file is created to implement the 5-fold cross-validation task. The pickles are loaded as X and y. The elements of X are image tensors. They are divided by 255 to put the inputs in the range of [0,1]. After that, the architecture of each model is called, which has been explained in the Models part.

4.1 Computing the Zero-One Loss

During each fold, the prediction for x-test is stored in the object named prediction. It is rounded and changed to a NumPy array with the name y-prediction-np. The sklearn has the methods accuracy-score and zero-one loss which allow calculating the accuracy and zero-one loss for each fold. The inputs of these two methods are NumPy arrays of "original labels" and the "predicted labels".

The accuracy and zero-one loss of each fold are stored in lists acc-model and loss-model respectively. In the end, the average of both accuracy and zero-one loss for the 5 folds are shown.

5 Results And Findings

5.1 Results

The results of the 5-fold cross-validation for different variants of the VGG16 model have been shown in Figure 3. The average loss of Model 1 is 22 percent, and its accuracy is around 77.8 percent. Model 2 whose optimizer is Adam, has a better performance in comparison with Model 1 by decreasing the loss by 2 percent. Model 4 has an additional block (Dense layer + Dropout layer)

in comparison with Model 2 and has much better performance. Model 4 has a zero-one loss of around 15.64 percent and its accuracy is 84.35 percent. Model 3 which has the stochastic Gradient Descent does not perform as well as Model 4 but it is still outperforming Model 1 and Model 2.

The results of the 5-fold cross-validation for different MobileNet models have been shown in Figure 4. The first version of MobileNet, Model 5, has a zero-one loss of 15.8 percent. Model 6 performs almost similarly to Model 5. Model 7 has a slightly better performance than Model 5 and Model 6. It demonstrates adding an additional block of the Dense layer and Dropout layer can generate better performance. Model 8 has the best performance by having just 6 percent of zero-one loss and an accuracy of 93.8 percent.

5.2 Research Findings

- Both CNN architectures VGG16 and MobileNet perform well when they face image sizes that are not in the standard size of (224,224). The image size for this work was just (64,64), which was highly smaller than the standard size.
- Considering the cross-validation, the results of 5-fold cross-validation for the 4 MobileNet models have significantly better performance than the 4 models of VGG16 architecture. The best performance of the VGG16 model is almost equal to the worse model in the MobileNet group. The zero-one loss of the best model in the MobileNet is around 10 percent less than the zero-one loss of the best model in the VGG16.
- In most models, the performance of the Adam optimizer was highly better than the performance of the Stochastic Gradient Descent when all the other parameters such as architecture, loss functions, number of epochs, etc. were the same.
- In most cases, the performance of the models with additional block layers (Dense+ Dropout) was higher than the performance of the models with the same condition without just having the additional block. This matter emphasizes the importance of customizing the pre-trained architectures such as VGG16 and MobileNet for the specific task, by adding additional layers.
- In most cases, the performance of the models with additional block layers (Dense+ Dropout) was higher than the performance of the models with the same condition without just having the additional block. This matter emphasizes the importance of customizing the pre-trained architectures such as VGG16 and MobileNet for the specific task, by adding additional layers.
- One of the goals of the task was to check how powerful are the VGG16 and MobileNet architectures with their saved weights. This research shows both architectures perform well in the new task which is different from their original task of multiclass classification of 1000 objects. Their saved weights could perform well for this binary classification task. This subject becomes more important when the limitation of the computational power of this research is also considered. In other words, the two CNN architectures performed well while the number of training parameters was limited and the computational abilities such as the number of epochs were also limited.

References

[1] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

```
156/156 [==============================] - 2s 11ms/step
 fold 1 has the accuracy of 0.7659318637274549: and loss of 0.23406813627254508
156/156 [==============================] - 2s 10ms/step
 fold 2 has the accuracy of 0.7712968530767689: and loss of 0.22870314692323113
156/156 [==============================] - 1s 8ms/step
 fold 3 has the accuracy of 0.7791140509120064: and loss of 0.22088594908799364
156/156 [==============================] - 1s 8ms/step
 fold 4 has the accuracy of 0.7955502104630187: and loss of 0.20444978953698134
156/156 [==============================] - 1s 8ms/step
 fold 5 has the accuracy of 0.781519342553618: and loss of 0.218480657446382
 The average of accuracy for 5 fold cross validation is: 0.7786824641465733
 The average of loss for 5 fold cross validation is: 0.22131753585342664
```

(a) Cross Validation Model 1

```
156/156 [==============================] - 114s 727ms/step
 fold 1 has the accuracy of 0.7853707414829659: and loss of 0.21462925851703407
156/156 [==============================] - 113s 724ms/step
 fold 2 has the accuracy of 0.79374624173181: and loss of 0.20625375826819003
156/156 [==============================] - 113s 722ms/step
 fold 3 has the accuracy of 0.7985568250150331: and loss of 0.2014431749849669
156/156 [==============================] - 113s 721ms/step
 fold 4 has the accuracy of 0.8157947484465825: and loss of 0.1842052515534175
156/156 [==============================] - 113s 721ms/step
 fold 5 has the accuracy of 0.7983563840448987: and loss of 0.20164361595510127
 The average of accuracy for 5 fold cross validation is: 0.798364988144258
 The average of loss for 5 fold cross validation is: 0.20163501185574195
```

+ Code + Markdown

(b) Cross Validation Model 2

```
156/156 [==============================] - 109s 699ms/step
 fold 1 has the accuracy of 0.7645290581162325: and loss of 0.2354709418837675
156/156 [==============================] - 104s 664ms/step
 fold 2 has the accuracy of 0.774103026658649: and loss of 0.22589697334135095
156/156 [==============================] - 107s 682ms/step
 fold 3 has the accuracy of 0.7823211064341551: and loss of 0.2176788935658449
156/156 [==============================] - 107s 682ms/step
 fold 4 has the accuracy of 0.7927440368811385: and loss of 0.20725596311886152
156/156 [==============================] - 107s 685ms/step
 fold 5 has the accuracy of 0.7881338945680497: and loss of 0.21186610543195028
 The average of accuracy for 5 fold cross validation is: 0.780366224531645
 The average of loss for 5 fold cross validation is: 0.219633775468355
```

(c) Cross Validation Model 3

```
156/156 [==============================] - 2s 10ms/step
 fold 1 has the accuracy of 0.8164328657314629: and loss of 0.1835671342685371
156/156 [==============================] - 2s 10ms/step
 fold 2 has the accuracy of 0.8258167969532972: and loss of 0.17418320304670276
156/156 [==============================] - 1s 8ms/step
 fold 3 has the accuracy of 0.8408498697133694: and loss of 0.1591501302866306
156/156 [==============================] - 1s 8ms/step
 fold 4 has the accuracy of 0.8586891160553217: and loss of 0.14131088394467828
156/156 [==============================] - 1s 9ms/step
 fold 5 has the accuracy of 0.8757265985167368: and loss of 0.12427340148326316
 The average of accuracy for 5 fold cross validation is: 0.8435030493940376
 The average of loss for 5 fold cross validation is: 0.1564969506059624
```


(d) Cross Validation Model 4

Figure 3: Cross-Validation of four VGG16 Models

```
156/156 [==============================] - 1s 5ms/step
 fold 1 has the accuracy of 0.8256513026052105: and loss of 0.17434869739478953
156/156 [==============================] - 1s 5ms/step
 fold 2 has the accuracy of 0.8376428141912207: and loss of 0.16235718580877934
156/156 [==============================] - 1s 5ms/step
 fold 3 has the accuracy of 0.8422529565043095: and loss of 0.15774704349569046
156/156 [==============================] - 1s 5ms/step
 fold 4 has the accuracy of 0.8476648626979355: and loss of 0.1523351373020645
156/156 [==============================] - 1s 5ms/step
 fold 5 has the accuracy of 0.8542794147123672: and loss of 0.14572058528763276
 The average of accuracy for 5 fold cross validation is: 0.8414982701422087
 The average of loss for 5 fold cross validation is: 0.15850172985779132
```

(a) Cross Validation Model 5

```
156/156 [==============================] - 1s 5ms/step
 fold 1 has the accuracy of 0.8304609218436874: and loss of 0.16953907815631264
156/156 [==============================] - 1s 6ms/step
 fold 2 has the accuracy of 0.8252154740428944: and loss of 0.17478452595710559
156/156 [==============================] - 1s 5ms/step
 fold 3 has the accuracy of 0.8320304670274604: and loss of 0.16796953297253958
156/156 [==============================] - 1s 5ms/step
 fold 4 has the accuracy of 0.8318300260573261: and loss of 0.16816997394267386
156/156 [==============================] - 1s 5ms/step
 fold 5 has the accuracy of 0.8386450190418921: and loss of 0.16135498095810785
 The average of accuracy for 5 fold cross validation is: 0.831636381602652
 The average of loss for 5 fold cross validation is: 0.1683636183973479
```

(b) Cross Validation Model 6

```
156/156 [==============================] - 1s 5ms/step
 fold 1 has the accuracy of 0.8258517034068136: and loss of 0.17414829659318642
156/156 [==============================] - 1s 5ms/step
 fold 2 has the accuracy of 0.8360392864301464: and loss of 0.16396071356985364
156/156 [==============================] - 1s 5ms/step
 fold 3 has the accuracy of 0.8422529565043095: and loss of 0.15774704349569046
156/156 [==============================] - 1s 5ms/step
 fold 4 has the accuracy of 0.8564842653838445: and loss of 0.14351573461615552
156/156 [==============================] - 1s 5ms/step
 fold 5 has the accuracy of 0.8618961715774704: and loss of 0.13810382842252955
 The average of accuracy for 5 fold cross validation is: 0.8445048766605169
 The average of loss for 5 fold cross validation is: 0.15549512333948312
```

(c) Cross Validation Model 7

```
156/156 [==============================] - 1s 5ms/step
 fold 1 has the accuracy of 0.8559118236472946: and loss of 0.14408817635270543
156/156 [==============================] - 1s 5ms/step
 fold 2 has the accuracy of 0.9011826017237924: and loss of 0.09881739827620761
156/156 [==============================] - 1s 5ms/step
 fold 3 has the accuracy of 0.9607135698536781: and loss of 0.03928643014632194
156/156 [==============================] - 2s 6ms/step
 fold 4 has the accuracy of 0.9821607536580477: and loss of 0.01783924634195233
156/156 [==============================] - 1s 5ms/step
 fold 5 has the accuracy of 0.9923832431348968: and loss of 0.007616756865103214
 The average of accuracy for 5 fold cross validation is: 0.9384703984035419
 The average of loss for 5 fold cross validation is: 0.06152960159645811
```

(d) Cross Validation Model 8

Figure 4: Cross-Validation of four MobileNet Models

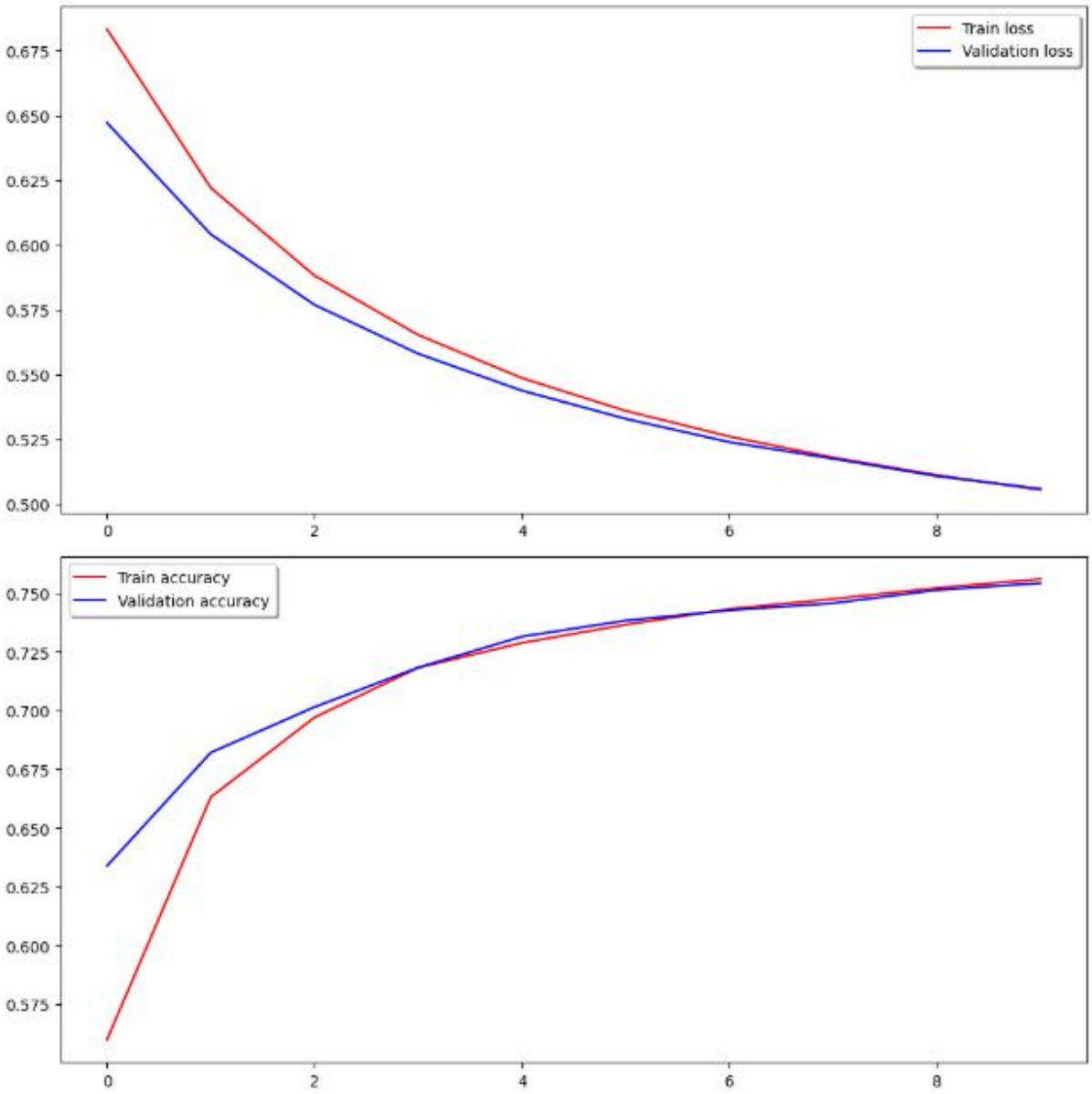


Figure 5: Model 1: Training VGG16 Base Model with SGD Optimizer

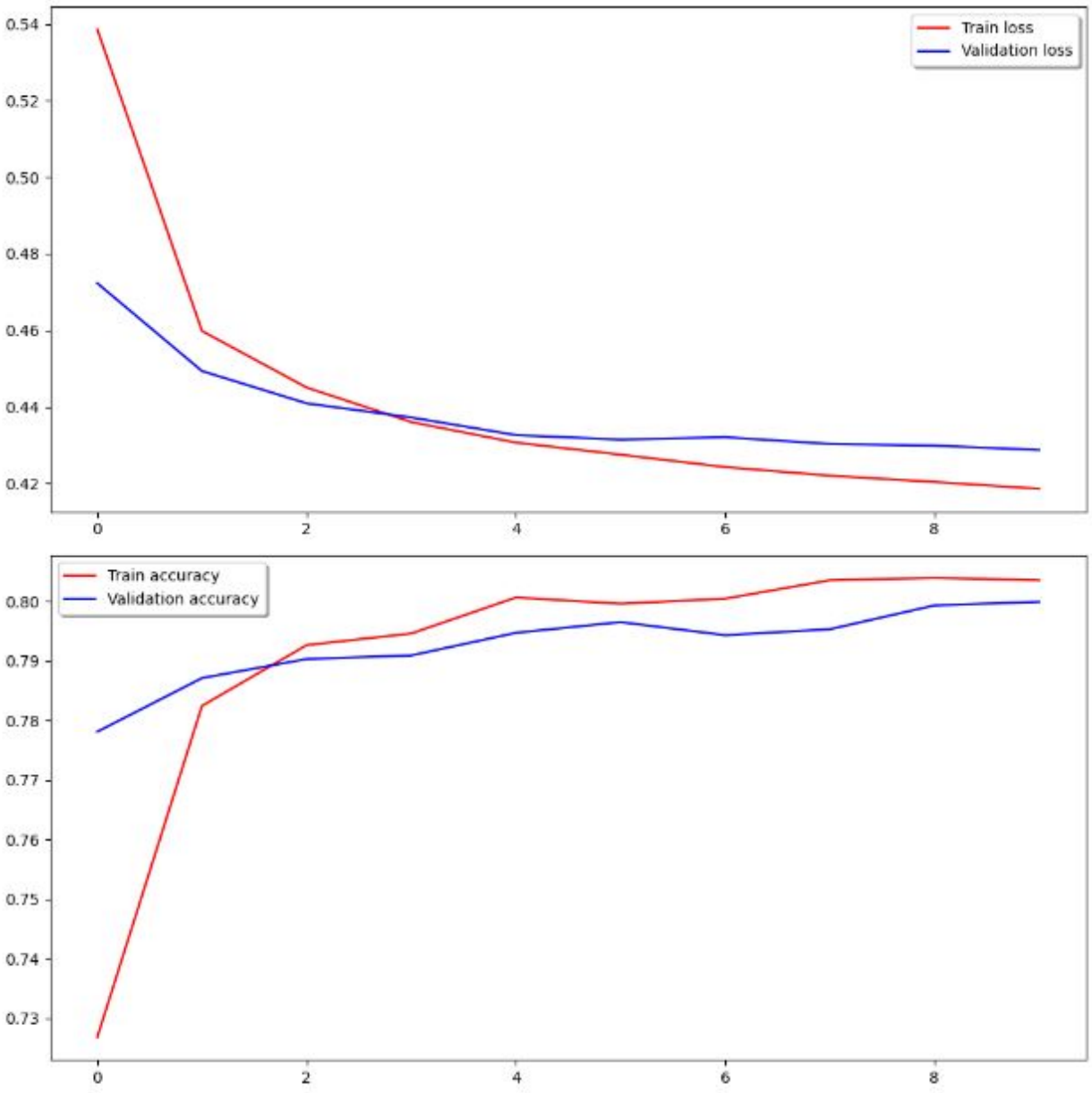


Figure 6: Model 2: Training VGG16 Base Model with Adam Optimizer

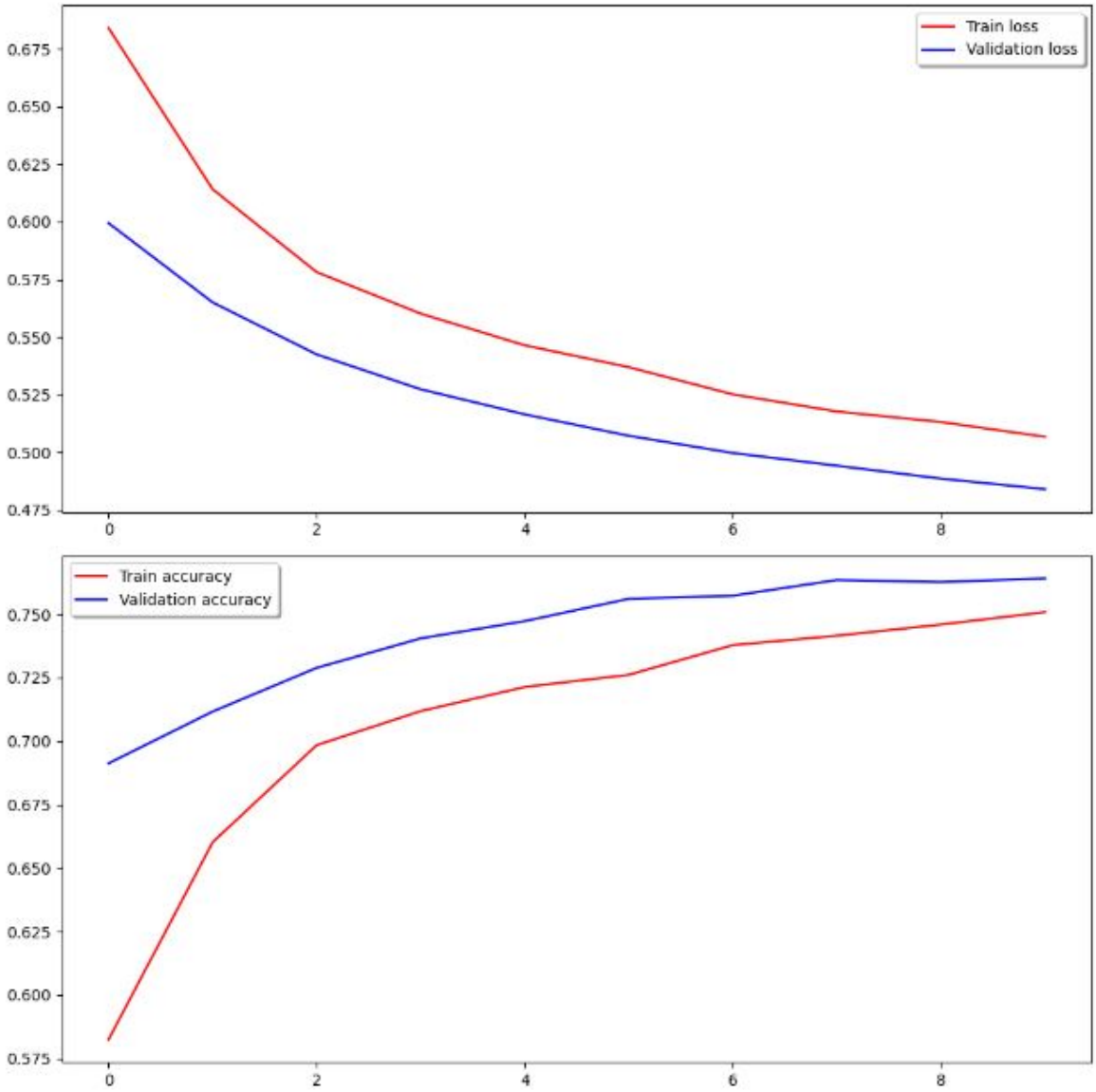


Figure 7: Model 3: Training VGG16 with Additional Block and with SGD Optimizer

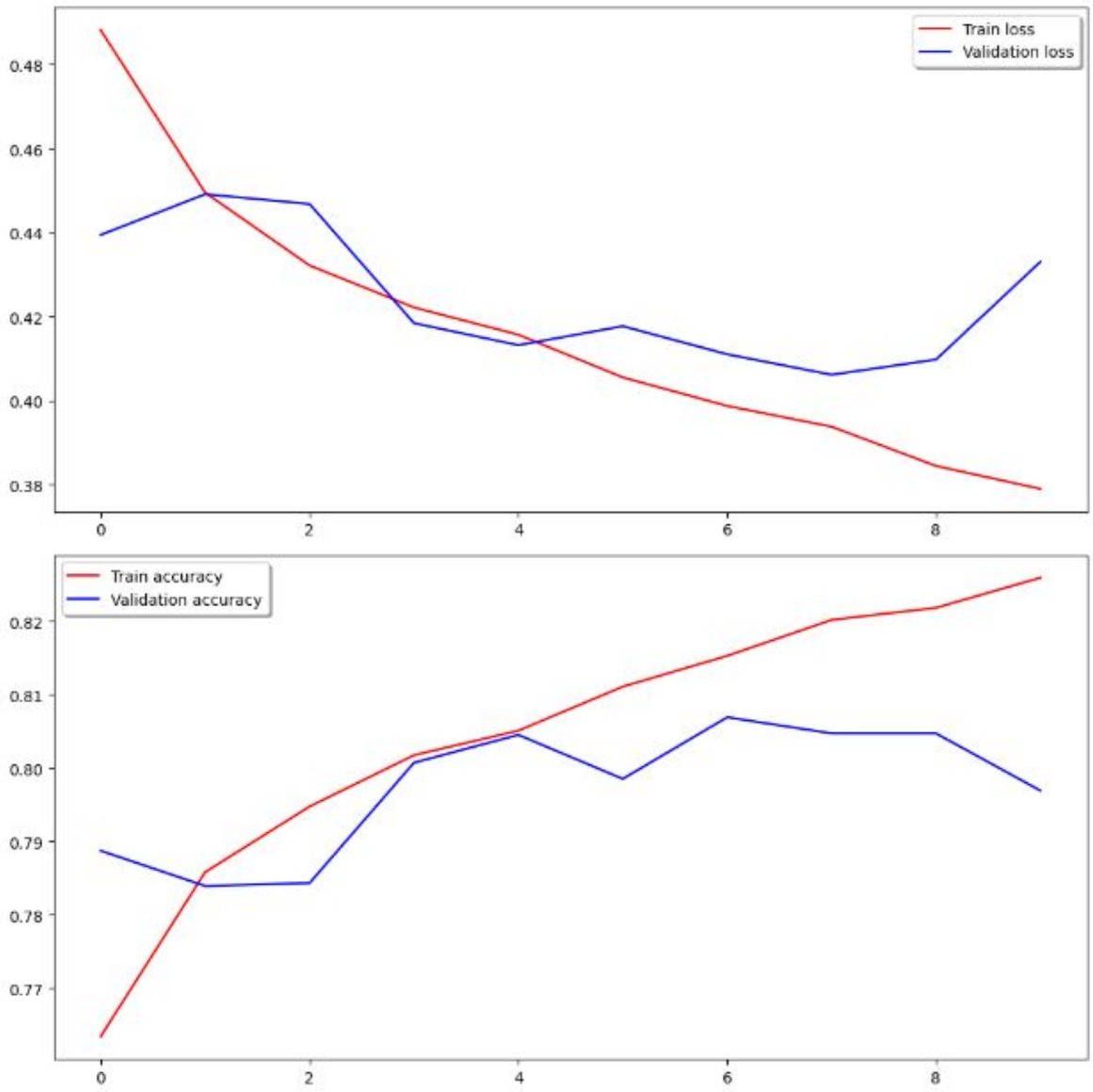


Figure 8: Model 4: Training VGG16 with Additional Block and with Adam Optimizer

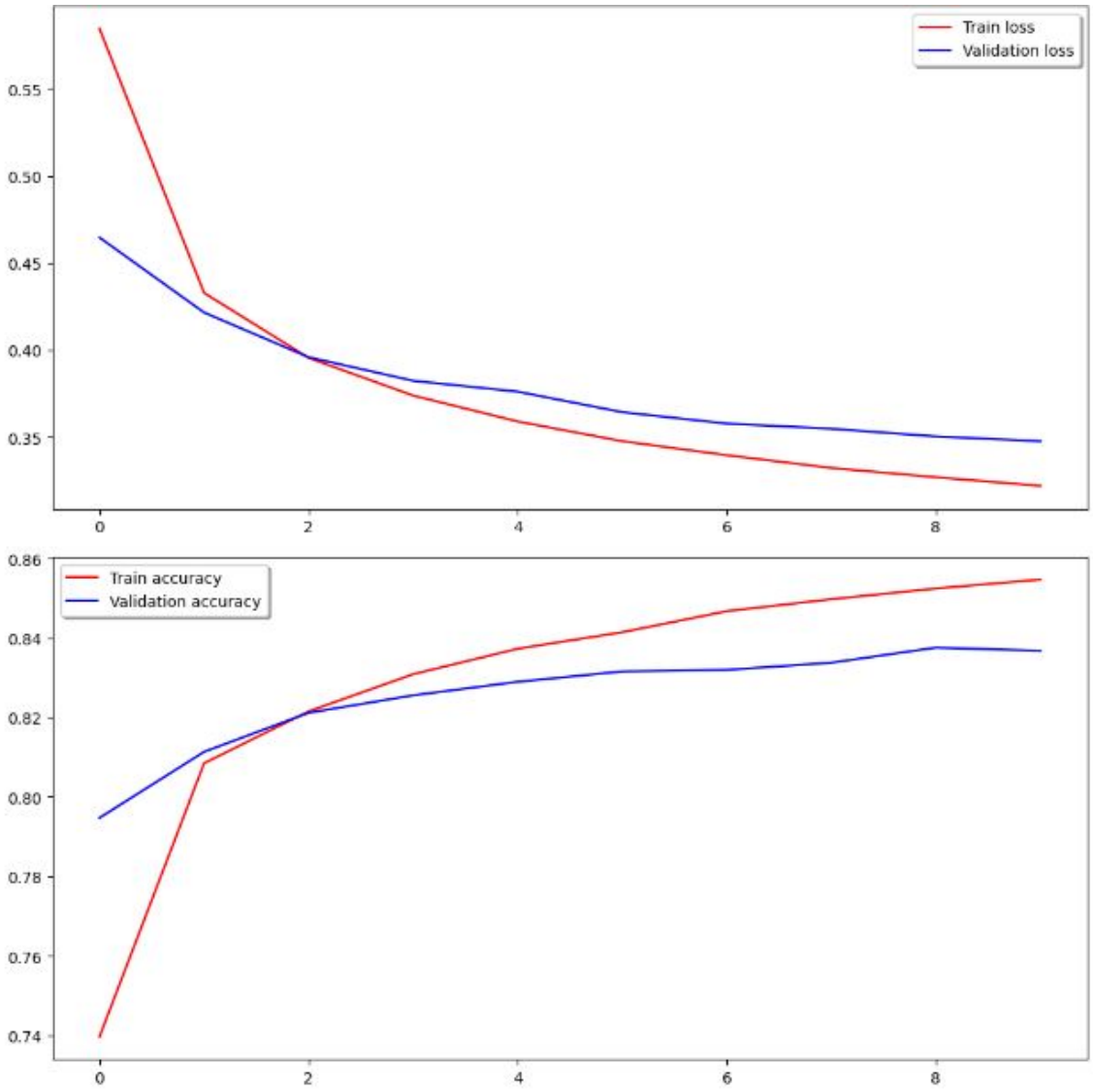


Figure 9: Model 5: Training MobileNet Base Model with SGD Optimizer

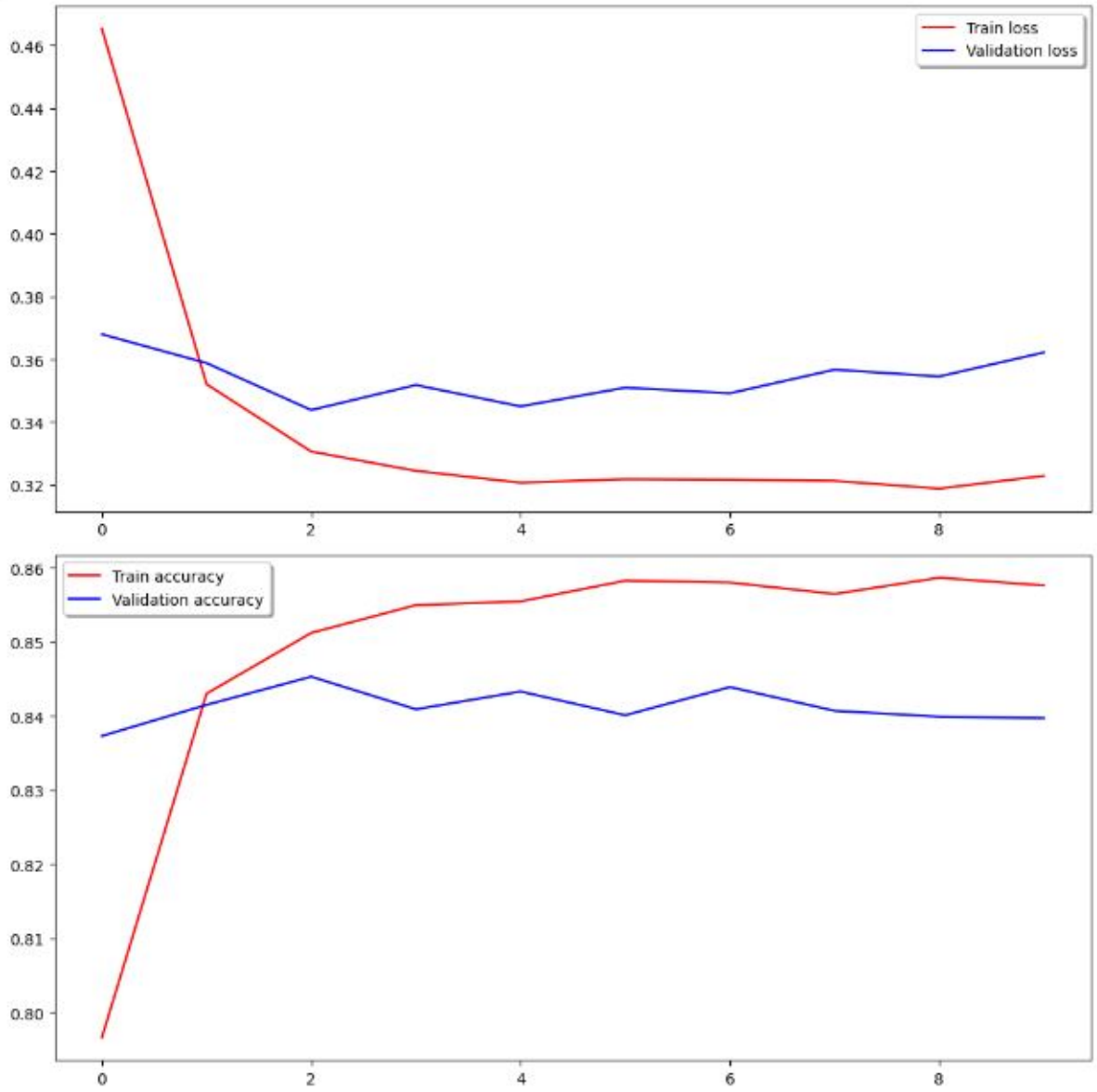


Figure 10: Model 6: Training MobileNet Base Model with Adam Optimizer

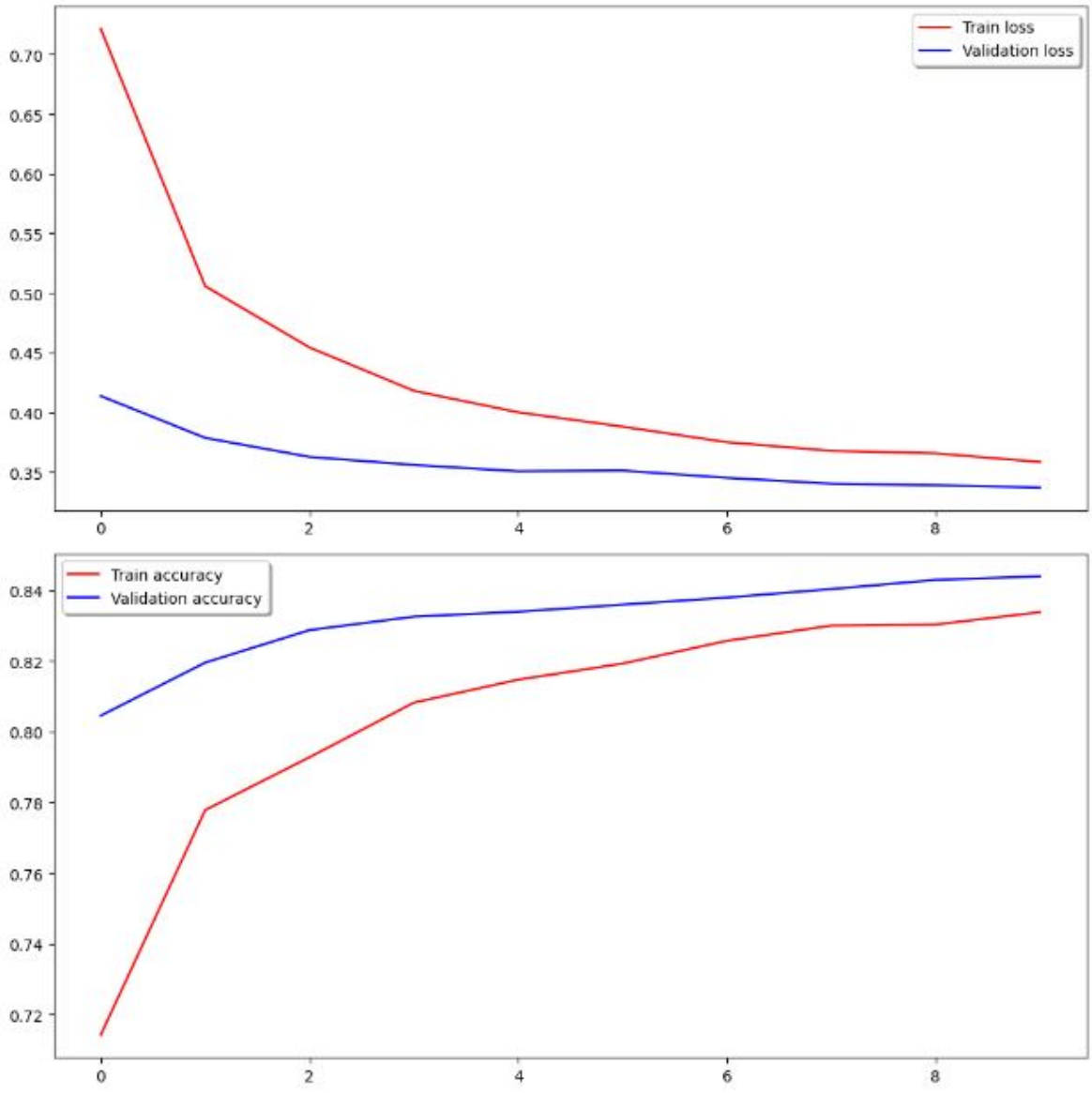


Figure 11: Model 7: Training MobileNet with Additional Block with SGD Optimizer

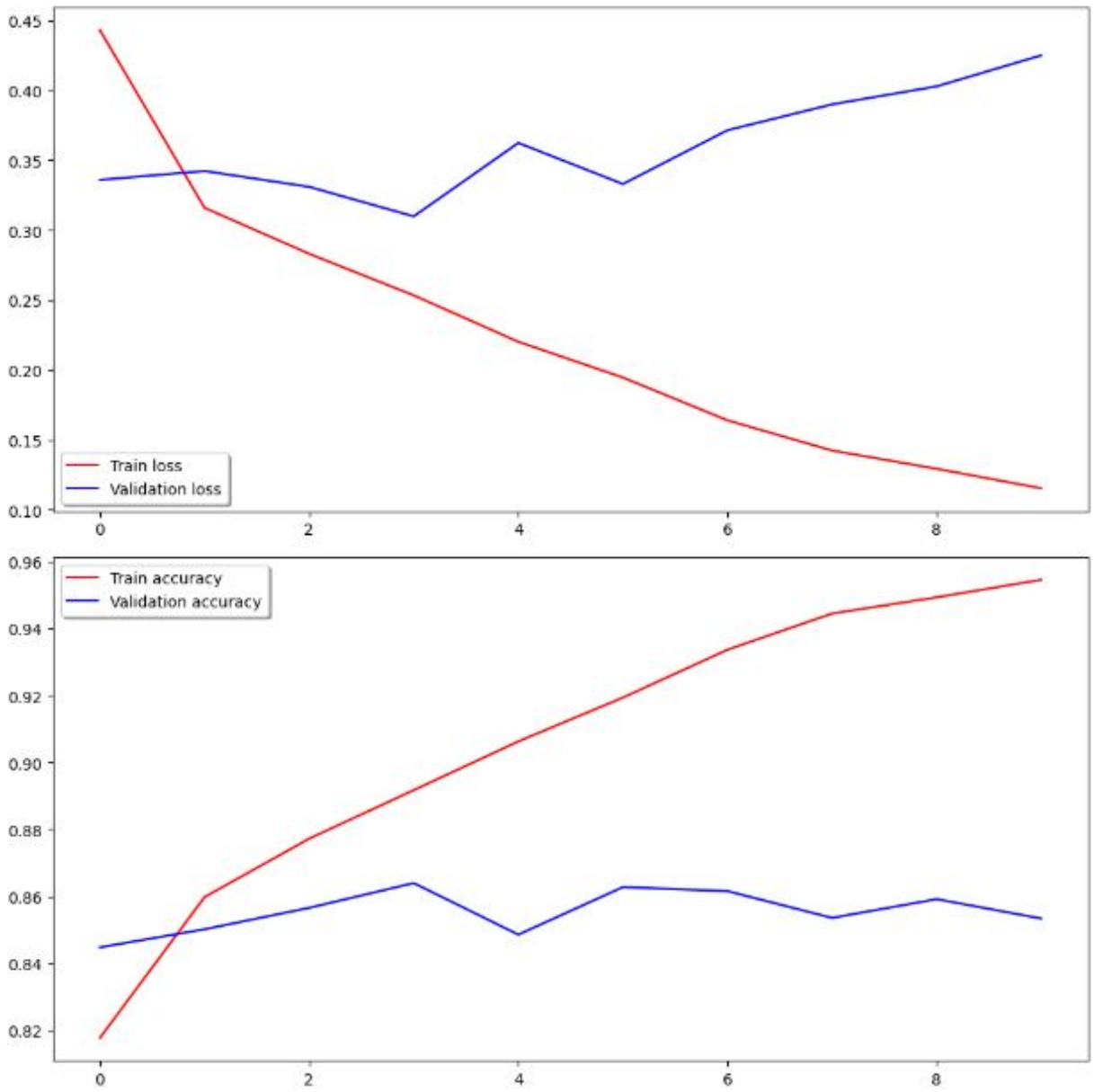


Figure 12: Model 8: Training MobileNet with Additional Block with Adam Optimizer

[2] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.