Multi-Class Image Classification with Deep Learning

MSDS 458 – Assignment 4

Siyuan Liu

Email: siyuanliu2022@u.northwestern.edu

March 11th, 2022

Abstract

Multi-class image classification has been one of application areas as a subset of computer vision domain where deep learning has achieved great progress in last decade together with object detection and image segmentation. As one of deep learning model structure, Convolutional Neural Network (CNN) has been proved to outperform other models especially in image classification because of its advantageous capability to extract features from the input images and evolve based on self-learning. In this research, the “Dogs vs. Cats” dataset, a collection of color images from Kaggle.com has been used to train and experiment with multiple deep learning models. The primary objective of this research is to experiment different hyperparameter tuning and model architecture at micro level on top of baseline models, and then explore some more advanced model architecture and algorithms such as ResNet50 and VGG16. Performance across models are put next to each other for comparison and key insights generation.

Key Words:

Multi-class image classification; Neural network; Deep learning; Convolutional Neural Network (CNN)

1. Introduction

Visual object recognition has been one of focused areas where a lot of artificial intelligence studiers attempt to train machine to mimic human being capability of detecting and recognizing images which vary in position, scale, pose and so on. Recently, deep neural networks have achieved impressive progress in this area, while CNN outperforms others due to its advantageous capability of extracting features from images through self-learning (Tien 2018). In the research, several experiments are conducted to experiment with multiple deep learning models on on “Dogs vs. Cats” dataset from one of competitions on Kaggle.com. The main object of conducting these experiments is to experiment different hyperparameter tuning, such as optimizers and regularization, and model architecture at micro level such as dropout layers, on top of baseline models that are built up on CNN or DNN architectures. Additionally, some more advanced model architecture and algorithms such as ResNet50, GAN, VGG 16 are explored to compare performance against the models that are built from scratch.

Literature review

Image classification is one of prevalent and fundamental areas where deep learning has proved to achieve impressive progress. The VGGNet has improved the model performance upon AlexNet structure since it won the Large Scale Visual Recognition Challenge in 2012 (Simonyan and Zisserman, 2014). GoogLeNet has improved the performance by enhancing (Szegedy, Liu, and Sermanet, 2015). ResNet developed by He and his team in 2016 has reduced the difficulty of training deep convolutional neural networks by introducing shortcut connections and residual representation during training. In 2017, DenseNet was proposed with each layer obtaining additional inputs from all preceding layers and passes on its own feature maps to all the subsequent layers (Huang et al, 2017)

2. Method

3.1. Data Collection & Exploration

The “Dogs vs. Cats” dataset provided by Kaggle consists of a set of .jpg format raw images including 25,000 training instances with labels and 12,500 testing instance without labels as shown in Figure 1. Both distributes evenly between two classes. The classes are completely mutually exclusive.

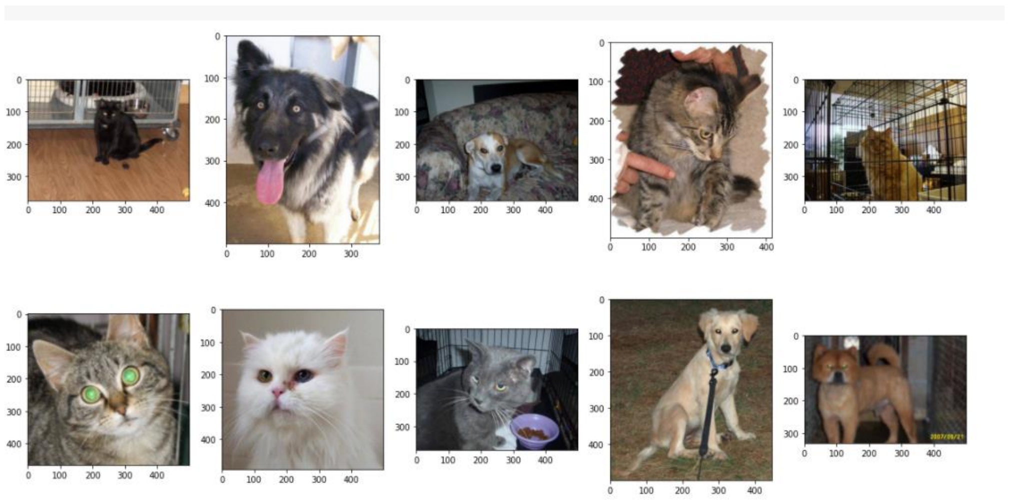


Figure 1. Dogs vs Cats Dataset Overview (Kaggle)

3.2 Data Preprocessing

The raw images are firstly processed through OpenCV package to converted into NumPy arrays in 4 dimensions. For the sake of computation complexity, 64 x 64 x 1 is picked as processing configuration to export as image height, width. Grayscale (1 channel) instead of color (3 channels) is set as channel used for most of model training except for ResNet50 and VGG16. For the pre-trained models including ResNet50 and VGG16, 224 x 224 x 3 arrays are generated with RGB channels for training using ImageDataGenerator embedded in TensorFlow as it is recommended input data configuration. For DNN models, data has been reshaped to flat array for training.

Data normalization is implemented necessarily before the image data is fed into classification training. Given all pixels are in color-scale ranging from 0 to 255, the data is normalized to 0 to 1 simply by dividing all pixel data by 255 since it is more favorable for neural network model computation.

3.3 Model Construction and Training

In this study, twelve neural network models have been constructed and experimented with diversified model architectures ranging from fundamental models built up on CNN or DNN layers from scratch and more sophisticated pre-trained models including VGG16 and ResNet50. Additionally, a Generative Adversarial Network (GAN) is explored to testify its discriminator on classification task within this study. A summary of all models covered is illustrated in Figure 2.

Model	**Model Structure**	**Number of Layers**	Optimizer	Padding	Regularization
Model 1	DNN	3	Adam	Valid	No regularization
Model 2	DNN	3	Adam	Valid	L2 Regularization (lr=0.001) Dropout (0.3)
Model 3	CNN	3	Adam	Valid	No regularization
Model 4	CNN	3	Adam	Valid	L2 Regularization (lr=0.001) Dropout (0.3) after Conv. Layer only Batch Normalization
Model 5	CNN	3	Adam	Valid	L2 Regularization (lr=0.001) Dropout (0.3) after Dense Layer only Batch Normalization
Model 6	CNN	3	Adam	Valid	L2 Regularization (lr=0.001) Dropout (0.1) after Conv. Layer Dropout (0.5) after Conv. Layer Batch Normalization
Model 7	CNN	3	RMSprop	Valid	L2 Regularization (lr=0.001) Dropout (0.1) after Conv. Layer Dropout (0.5) after Conv. Layer Batch Normalization
Model 8	CNN	3	Adam	Same	L2 Regularization (lr=0.001) Dropout (0.1) after Conv. Layer Dropout (0.5) after Conv. Layer Batch Normalization
Model 9	CNN	5	Adam	Same Kernel size = 5 for 1st Conv.	L2 Regularization (lr=0.001) Dropout (0.1) after Conv. Layer Dropout (0.5) after Conv. Layer Batch Normalization
Model 10	ResNet50	50	Adam	Fixed padding	/
Model 11	VGG16	16	Adam	Same	Dropout (0.5) after last Conv. Layer Dropout (0.5=3) after Dense Layer
Model 12	GAN	1 supervised discriminator for classification purpose, 1 semi discriminator, 1 generator			

Figure 2. Model Architecture Overview

Since one of goals of this study is to get better understanding at more granular level on how micro tunning of model architecture would impact the performance, multiple baseline models are developed as benchmark. Two DNN models (Model 1 and 2) are built as referential models to compare with CNN models through Model 3 to 8, which are a series of CNN models with 3 layers convolutional layers as skeleton and applied with different tunning strategy on Dropout layer insertion location and parameters, Optimizer, and Regularization of last Dense layer. For example, Model 4, 5, and 6 experiment with inserting Dropout layer into different location of the model, Model 4 inserts Dropout with 0.3 ratio after the convolution layer only, Model 5 inserts Dropout layer with 0.5 ratio after the dense layer only, while Model inserts Dropout layer after both convolution layer and dense layer. Difference between model architecture has been demonstrated in Figure 3. The dropout ratio for the former is tweaked to 0.1 to maximumly reserve the information extracted from previous convolution layer but at the same reduce some overfitting factors.

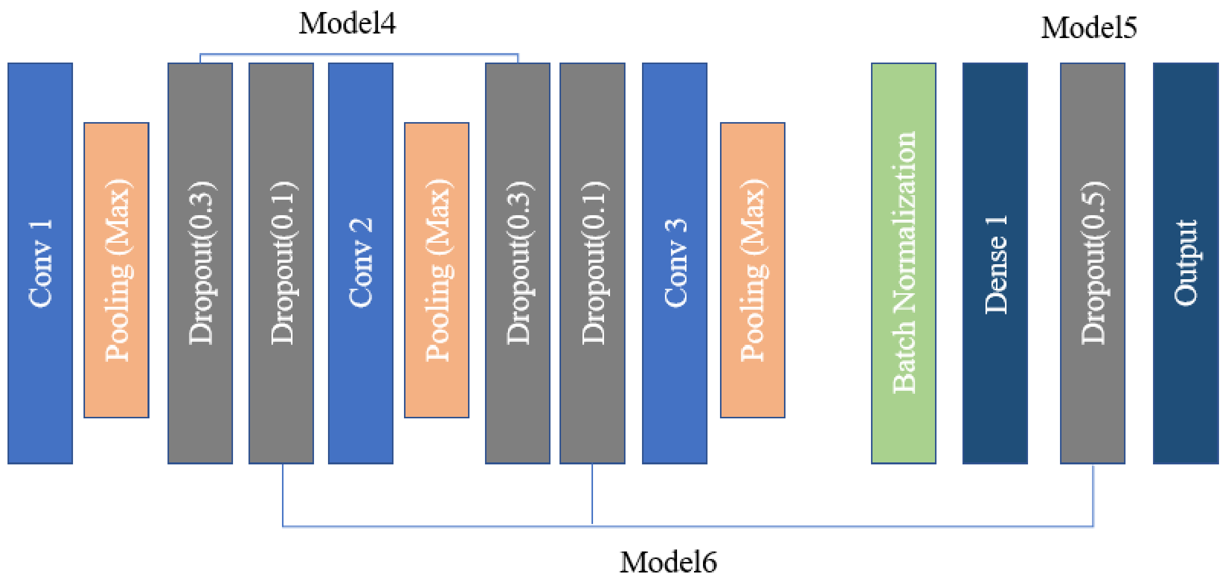


Figure 3. Model Architecture Difference (Dropout Layer Insertion) – Model 4/5/6

Model 7 swaps the "Adam" with "RMSprop" as optimizer on the top of same structure of Model 6. Model 8 tweaks the padding from the default "Valid" to "Same" in the convolution layers of Model 6 to observe how padding pattern impacts the model performance. Model 9 adds and stacks the convolutional layers (2nd and 3rd, 4th and 5th), and increase the kernel size to 5x5 from previous 3x3 to see how this change the feature extraction consequently impacts the model performance. Output of filters by different layers are plotted out and visualize how features are extracted and highlighted.

Model 10 and 11 explore pre-trained model structures including VGG16 and ResNet50 to basically get an idea on whether/how much stacked more layers could help improve the performance. In Model 12, a generative adversarial network (GAN) is developed with a supervised discriminator for classification task, a semi-supervised discriminator to identify fake versus real image, and a generator that produces fake images for exploration purpose.

3. Results

Experimental models are evaluated with performance metrics being traced, including accuracy, F1 score, recall, precision and computing time, as summarized in Figure 3.

Model	Accuracy train	Accuracy valid	F1 Score Valid	Recall Valid	Precision Valid	Training Time (s)
Model 1	0.6157	0.5922	0.5806	0.5932	0.6052	23.15
Model 2	0.6125	0.6024	0.6023	0.6025	0.6026	23.67
Model 3	0.9919	0.8318	0.8316	0.8316	0.8327	3379.88
Model 4	0.9107	0.7992	0.7948	0.8001	0.8297	4238.57
Model 5	0.7580	0.7130	0.6925	0.7145	0.7951	2536.94
Model 6	0.9817	0.8668	0.8667	0.8667	0.8670	4846.75
Model 7	0.9754	0.8366	0.8363	0.8363	0.8395	4828.66
Model 8	0.7437	0.7040	0.7056	0.7056	0.7979	2562.55
Model 9	0.9205	0.8490	0.8485	0.8550	0.8494	10608.52
Model 10	0.7822	0.7652	0.7647	0.7680	0.7655	2838
Model 11	0.864	0.7873	0.7543	0.7590	0.7428	Still Running
Model 12	Best classifier accuracy 66.55%					

Figure 4. Summary Table of Performance Score

As expected, CNN models outperform DNN models on image classification tasks as all other CNN models achieved above 70% accuracy in validation phase, while DNN models (Model 1 and 2) achieved 59.22% and 60.24% accuracy only.

Some findings could be drawn when we have a deep dive into Model 3, 4, 5 and 6. Model 3 reflects significant overfitting problem evidenced with gap between training accuracy 99.19% versus validation accuracy 83.18%, suggesting some regularization need to be set up to reduce overfitting as shown Figure 5.

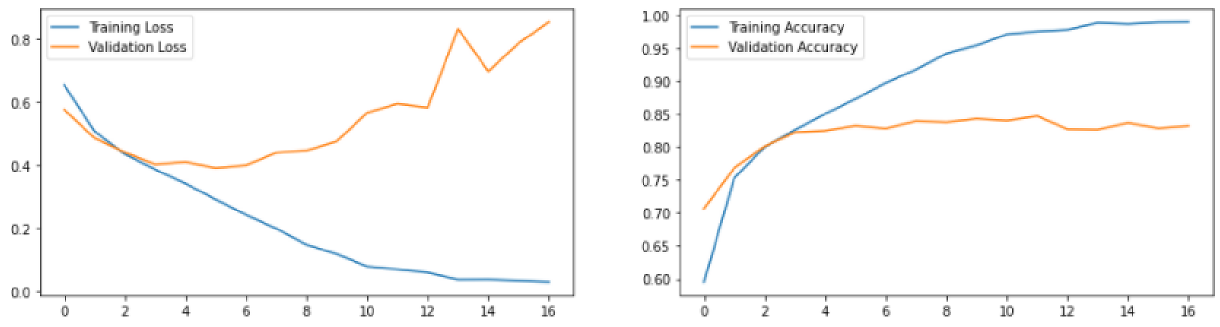


Figure 5. Performance Report of Model 4 (CNN 3 layers with no regularization)

As Dropout layers and L2 Regularization are brought in Model 4, 5, and 6, the overfitting problem has been mitigated. Model 4 achieves 91% training accuracy and 79.92% validation accuracy suggesting insertion of Dropout layer after the convolutional layer might cause information lost from the feature extraction leading to accuracy loss for both training and validation. Model 5 shrinks the gab between training accuracy 75.8% and validation accuracy 71.3% but both have declined significantly compared with Model 4. Model 6 achieves both higher training accuracy 98.17% and validation accuracy 86.68% after applying Dropout on both convolutional layer and dense layer. This is interesting because dropout is commonly used to regularize and be applied on fully-connected dense layers as proposed by Hinton in his original paper. There has been some controversial discussion on whether dropout should be applied on convolution layer, the performance gain in Model 6 suggests that there is some value by applying lower level (p-0.1 or 0.2) to the convolutional layers after the activation function. Model 7 with RMSprops achieves 97.54% and 83.66% for training and validation accuracy, beat by Model 6 with Adam as optimizer. This could be Adam or Adaptive Moment Optimization algorithms combines the heuristics of both Momentum and RMSProp, while RMSProp impedes the search in direction of oscillations. Model 8 uses “Same” padding for the convolutional layer but does not achieve higher accuracy than Model 6 with “Valid” padding, suggesting that there is little information at the border of image that are decisive for the classification. Model 9 achieves 92.05% training accuracy and 84.9% validation accuracy, which is second highest score. The model added two more convolutional layers with “Same” padding and converged at 17 epochs as its performance by epoch is plotted in Figure 6. It outperforms Model 8 with approximately 18% training accuracy gain and

14% validation accuracy gain suggesting the benefits of performance improvement brought by added two convolutional layers. Figure 7 demonstrates the output of filters from convolutional layers of Model 9.

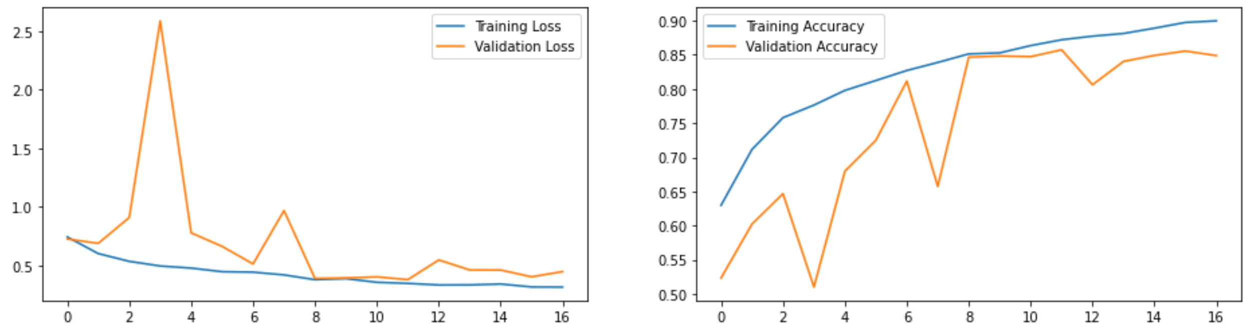
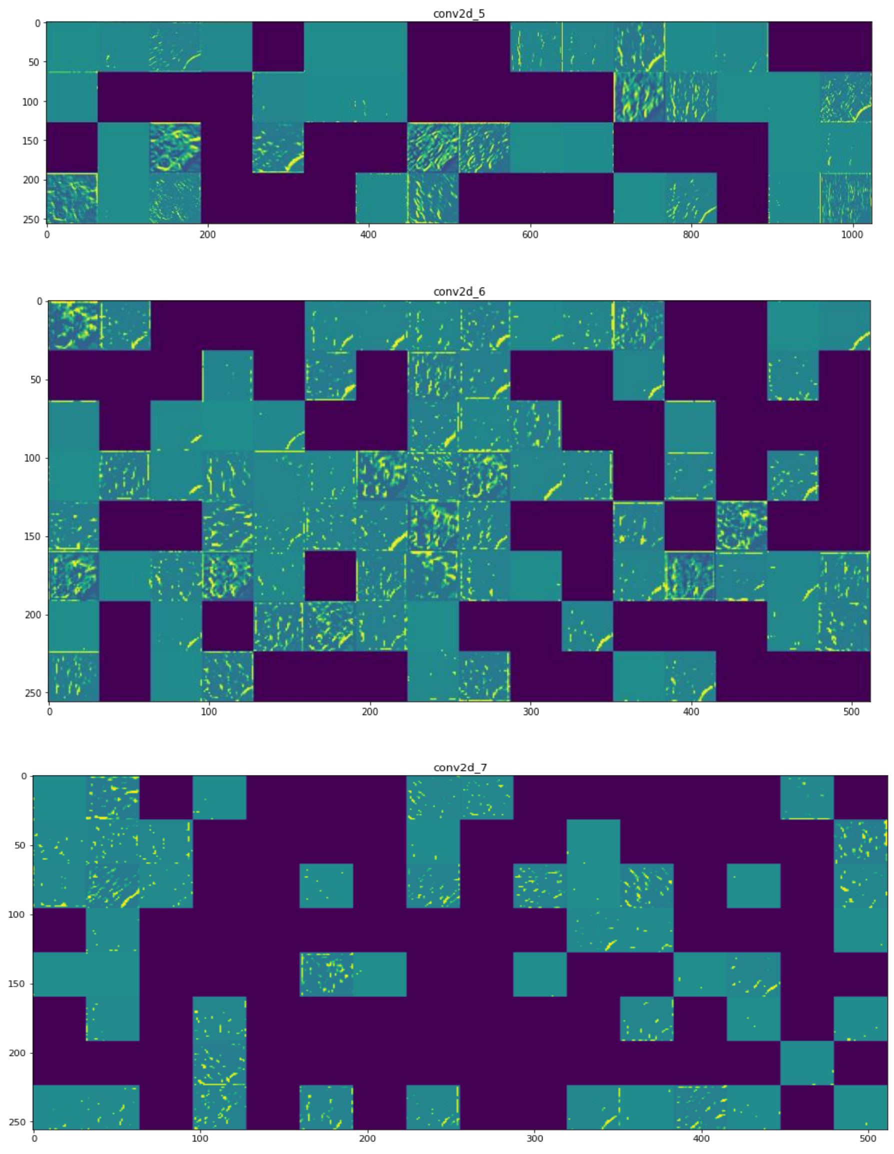
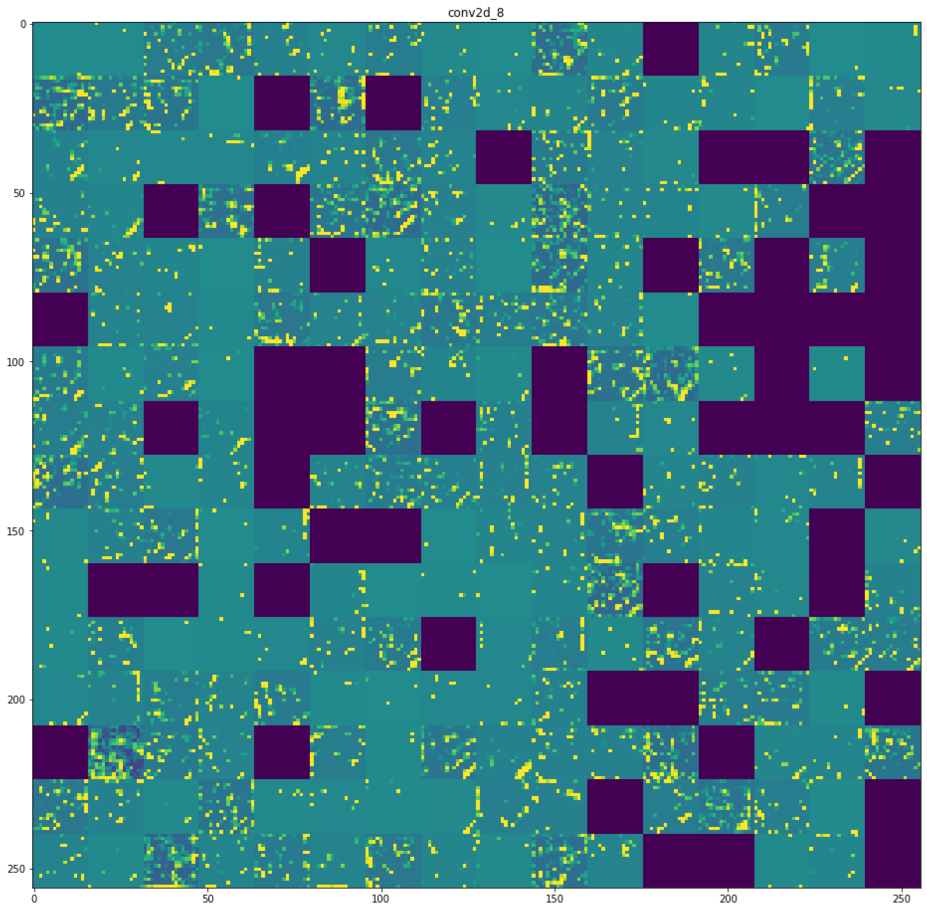


Figure 6. Performance Plot of Model 9 (CNN with 5 Conv. Layers and "Same" padding)





Model 10 leverages pre-trained ResNet50 for the training and achieves 78.22% training accuracy and 76.52% validation accuracy and 76.42% F1 score, which does not perform as good as expected. This is possibly because the training data is only 64 x 64 resolution, which is lower than the default configuration 224 x 224. Same does it for Model 11 VGG16, the training and validation accuracy is 86.4% and 78.73. For exploration purpose, a GAN model (Model 12) has been built to leverage its discriminator for classification. The best classifier accuracy it achieved is 66.55% by the time this report is submitted. The model is still running as accuracy is improved with generator and discriminator converge to equilibrium.

4. Conclusions

Overall, CNN outperforms DNN specifically in image classification field thanks to its extraordinary capability to extract global and local features based on self-learning. Dropout could be applied either to convolutional layer or dense layer to reduce overfitting problem, but certain attention is needed to be paid on dropout ratio to avoid possible information loss. Lower level (0.1 – 0.3) of dropout for convolution layer whereas the bar could be lifted to 0.5 for dense layer. Padding pattern are two other hyperparameters that could be fine-tuned looking for potential performance gain dependent on how much information is contained at the border of image for classification decision making. Optimizer is another factor to look into as their embedded algorithm will impact the convergence and performance dependent on its applicability to specific model architecture. Adding model complexity does not necessarily bring performance improvement to the model but rather possibly make the overfitting problem deteriorate as well as computational cost. How much capability the known pre-trained models such as ResNet50 and VGG series are dependent on other fundamental factors such as training data resolution and hyperparameters.

Some areas that are worth further investigation for this study is to increase resolution of training data from current 64 x 64 to higher configuration such as 224 x 224 or 256 x 256, so it might be more trainable for models with higher complexity. Also, data augmentation tactics could be applied to investigate its impact on performance improvement.

Reference:

Ho-Phuoc, Tien. “CIFAR10 to Compare Visual Recognition Performance Between Deep Neural Networks and Humans.” (2018): n. pag. Print.

Krizhevsky, A., Sutskever, I. and Hinton, G., 2017. ImageNet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6), pp.84-90.

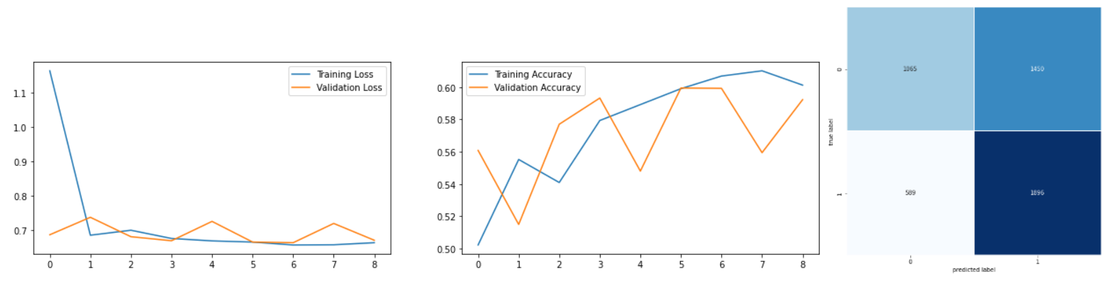
Han, Y., Zhang, P., Zhuo, T., Huang, W. and Zhang, Y., 2018. Going deeper with two-stream ConvNets for action recognition in video surveillance. *Pattern Recognition Letters*, 107, pp.83-90.

K. He, X. Zhang, S. Ren, and J. Sun, Deep residual learning for image recognition, *IEEE conference on computer vision and pattern recognition*, vol. 7, 2016, pp. 770–778.

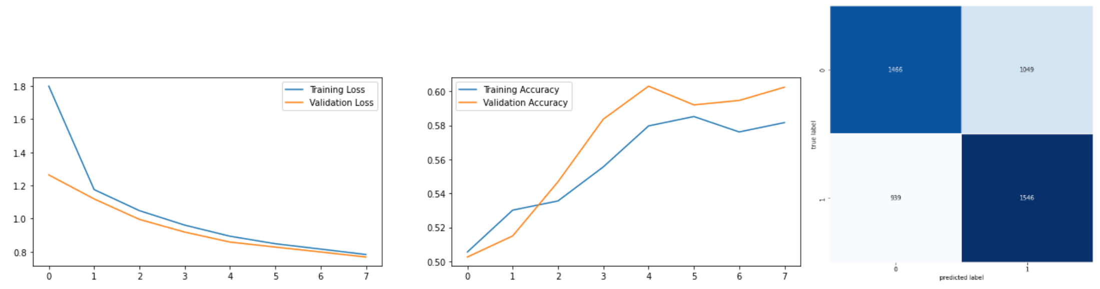
G. Huang, Z. Liu, L. Van Der Maaten, K. Weinberger. Densely connected convolutional networks, *IEEE Conference on Computer Vision and Pattern Recognition*, 2017 (2017), pp. 2261-2269

Appendix

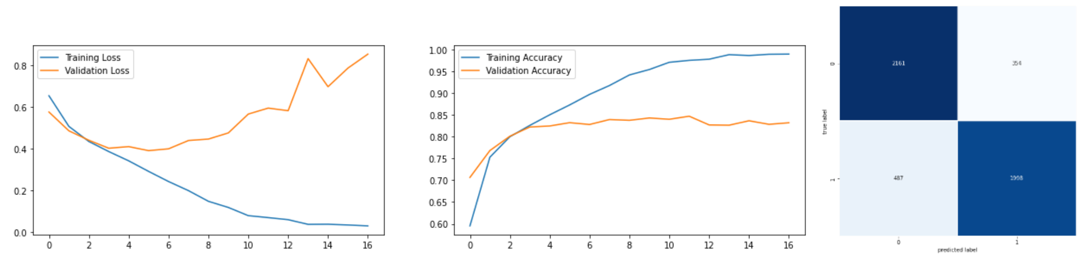
Model 1 Performance Plot



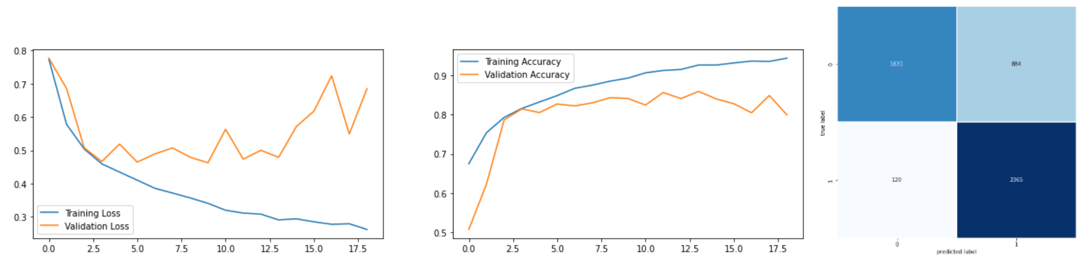
Model 2 Performance Plot



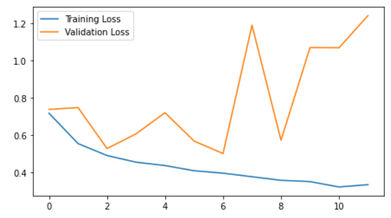
Model 3 Performance Plot

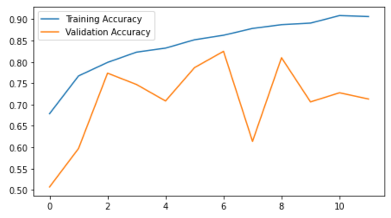


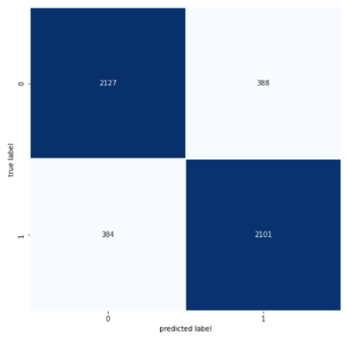
Model 4 Performance Plot



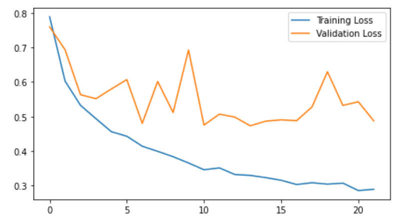
Model 5 Performance Plot

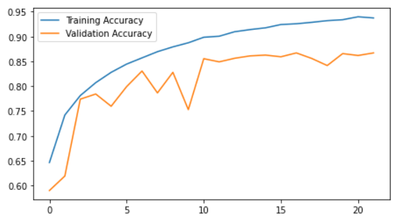


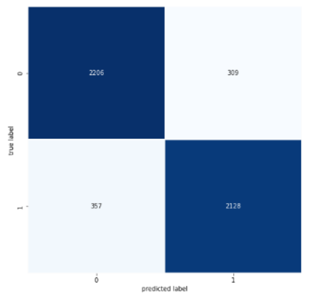




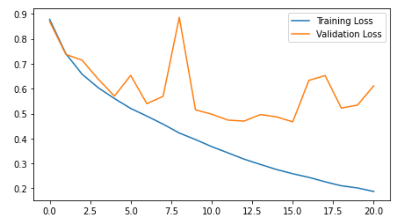
Model 6 Performance Plot

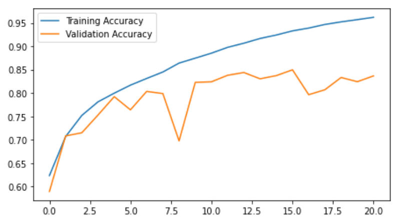


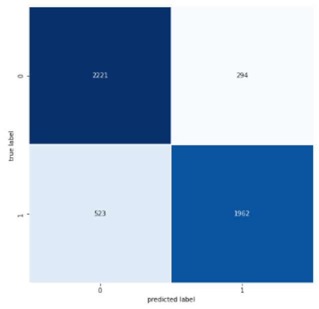




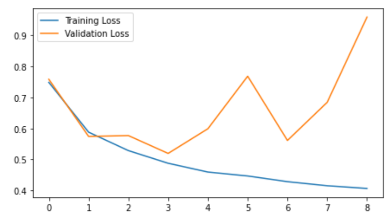
Model 7 Performance Plot

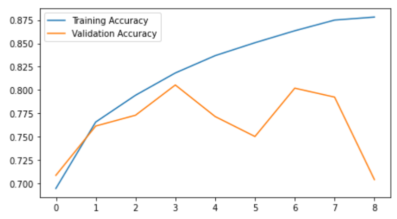


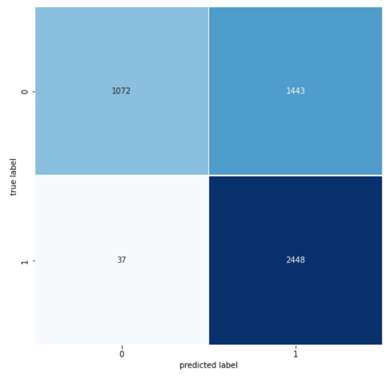




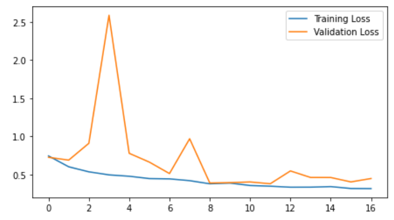
Model 8 Performance Plot

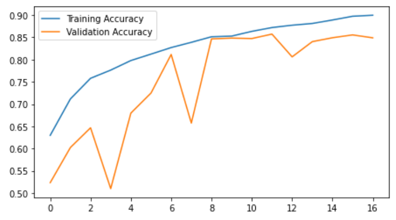


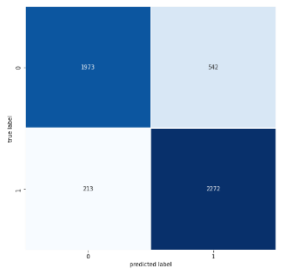




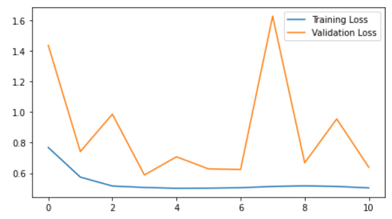
Model 9 Performance Plot

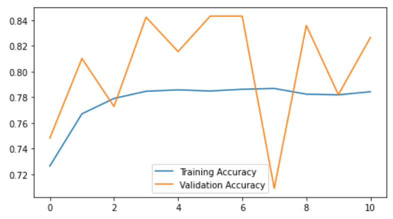


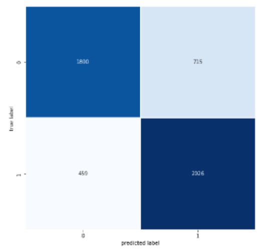




Model 10 Performance Plot


Training Loss
Validation Loss


Training Accuracy
Validation Accuracy


true label
predicted label