CIFAR10 CNN use Keras

Xinyi Li

December 2018

1. Data Exploration

1-1 CIFAR 10 description

- **data** -- a 10000x3072 numpy array of uint8s. Each row of the array stores a 32x32 color image. The first 1024 entries contain the red channel values, the next 1024 the green, and the final 1024 the blue. The image is stored in row-major order, so that the first 32 entries of the array are the red channel values of the first row of the image.
- **labels** -- a list of 10000 numbers in the range 0-9. The number at index i indicates the label of the ith image in the array data. The dataset contains another file, called batches.meta. It too contains a Python dictionary object. It has the following entries:
- **label_names** -- a 10-element list which gives meaningful names to the numeric labels in the labels array described above. For example, label_names[0] == "airplane", label_names[1] == "automobile", etc.

1-2 CIFAR 10 Exploration

Figure 1: Random image of each class

1-3 CIFAR 10 Data Processing

- rotation_range: Int. Degree range for random rotations.
- width_shift_range: Float (fraction of total width). Range for random horizontal shifts.
- height_shift_range: Float (fraction of total height). Range for random vertical shifts.
- horizontal_flip: Boolean. Randomly flip inputs horizontally[2].

Figure 2: Original images and Augmented images

2. Convolutional Neural Networks (CNN) with Keras

Introduction for Keras

- Keras is an open source neural network library written in Python.
- Keras contains numerous implementations of commonly used neural network building blocks such as layers, objectives, activation functions, optimizers, and a host of tools to make working with image and text data easier[2].
- It is easy to get started with Keras and the results is much more readable.

2-1 Model 1: Simple model

Goal

To start with, we build a simple CNN classifier from Keras and explore details to understand how convolution works. Comparing the different batch size (32, 64, 128) and choosing one batch size number for the following models.
We trained the model with 100 epochs.

Architecture

Layer (type)	Output Shape	Param #
conv2d_1 (Conv2D)	(None, 30, 30, 32)	896
max_pooling2d_1 (MaxPooling2	(None, 15, 15, 32)	0
conv2d_2 (Conv2D)	(None, 13, 13, 64)	18496
max_pooling2d_2 (MaxPooling2	(None, 6, 6, 64)	0
flatten_1 (Flatten)	(None, 2304)	0
dense_1 (Dense)	(None, 10)	23050

Total params: 42,442
Trainable params: 42,442
Non-trainable params: 0

Figure3: Architecture of Model 1

- **Convolutional layer:** CNN are used to recognize images by transforming the original image through layers to a class score.
- **Max pooling**: Its function is to progressively reduce the spatial size of the representation to reduce the amounts of parameters and computation in the network. Pooling layer operates on each feature map independently[3].
- **Flatten layer**: When you finish editing all the layers in your image, you can merge or flatten layers to reduce the file size. Flattening combines all the layers into a single background layer.
- **Dense layer**: dense layer is simply a layer where each unit or neuron is connected to each neuron in the next layer.

I use the Adam algorithm to optimize the weights on the backpropagation, set optimizer='adam'.
Adam, an algorithm for first-order gradient-based optimization of stochastic objective functions, based on adaptive estimates of lower-order moments. The method is straightforward to implement, is computationally efficient, has little memory requirements, is

invariant to diagonal rescaling of the gradients, and is well suited for problems that are large in terms of data and/or parameters.

Results

1. Parameter setting and result comparison

	Model 1-1	Model 1-2	Model 1-3
batch size	32	64	128
epoch	100	100	100
test loss	2.2219	1.9812	1.5645
test accuracy	0.662	0.6716	0.6834

Table 1: Results summary of Model 1

Choose batch size 128.

2. Tran/Test Accuracy and Loss

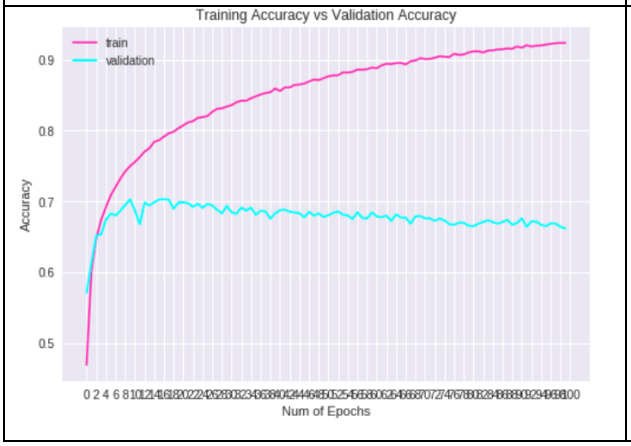
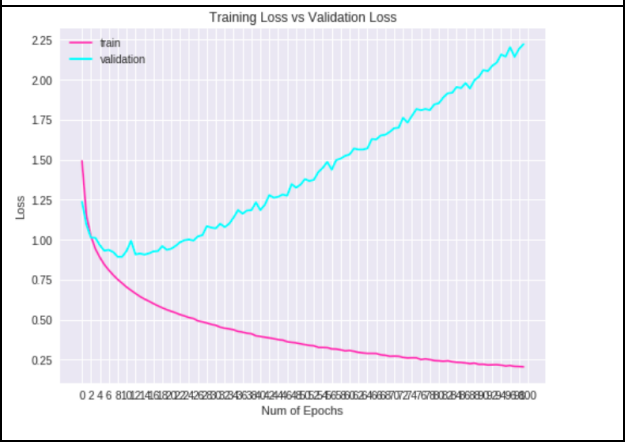
Accuracy	Loss
	

Table 2: Model 1-1 batch size 32

Accuracy	Loss

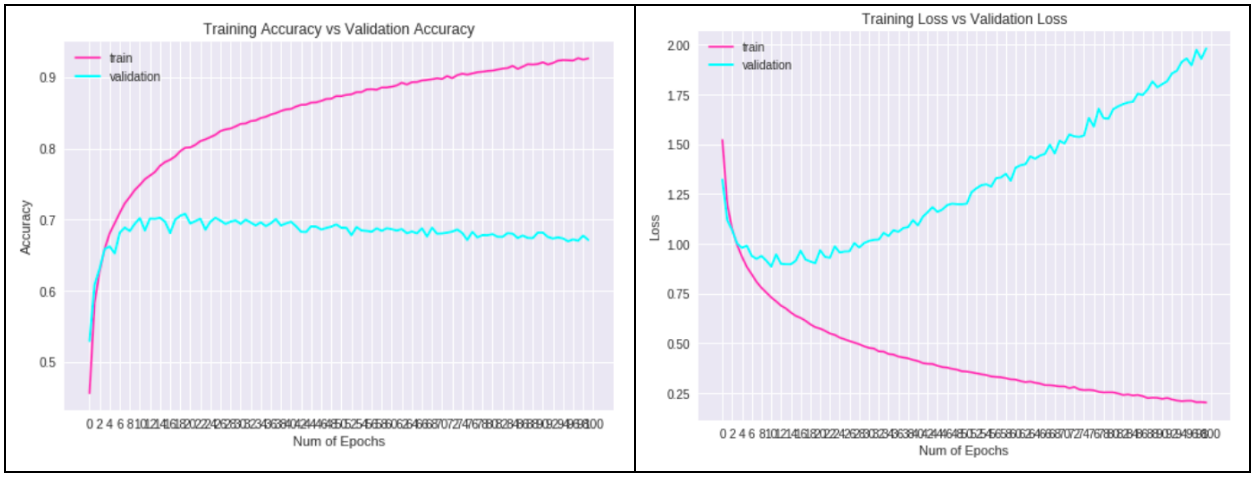


Table 3: Model 1-2 batch size 64

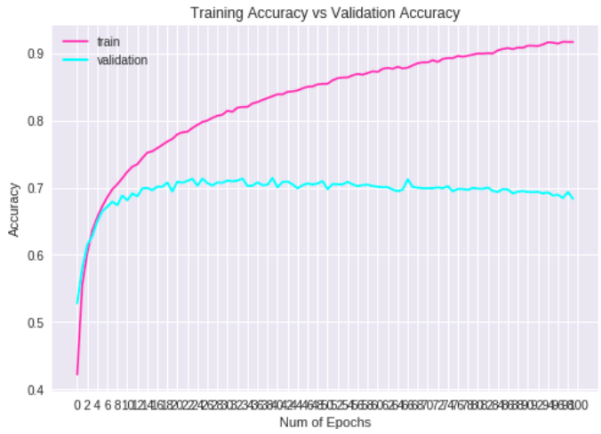
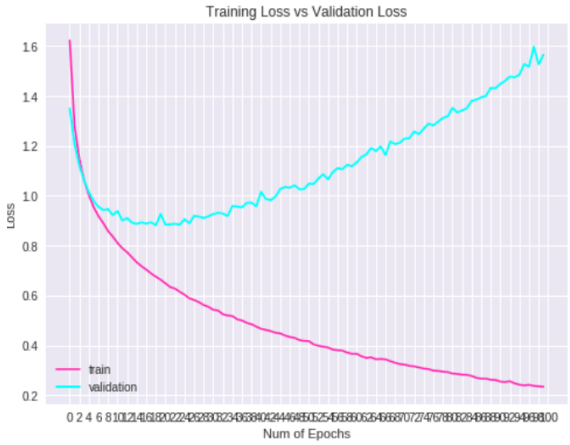
Accuracy	Loss
	

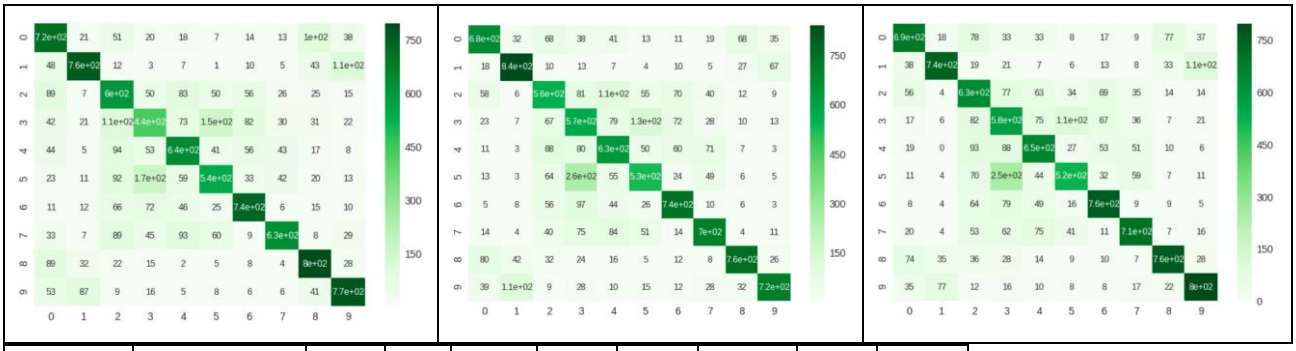
Table 4: Model 1-3 batch size 128

We train the model for 100 epochs with three different batch size (32, 64, 128).
After 100 epochs, the accuracy on test dataset is around 68.34% in model 1-3, so we set batch size equal to 128.

3. Confusion Matrix

We use confusion matrix to see which of the classes are more likely to be misclassified.

Model 1-1	Model 1-2	Model 1-3



0	1	2	3	4	5	6	7	8	9
airplane	automobile	bird	cat	deer	dog	frog	horse	ship	truck

Table 5: Confusion matrix result of Model 1

The darker the color (at the same time, the bigger the number), the less likely it is to misclassify. For instance, in Model 1-3 the darkest is the ninth, which means truck are less likely to be misclassified, whereas 48% of dogs are given the wrong labels. 25% of dogs are misclassified as cat. Although dogs are usually classified wrongly, they are often mistaken as animals instead of objects.

4. Examples of Mis-classified Images

Figure 4: Right classified and Mis-classified images in Model 1-3

Above all are 32 examples of images that are predicted wrong by Model 1-3 CNN classifiers. Some of the images are easy for humans to detect the difference but hard for the CNN classifiers. For example, the second one in the last row is easy to identity it's an automobile but mis-classified as a cat by the CNN classifiers. This means that the classifier makes some simple mistakes, and there are many places that need improvement.

Insights:

The test result is 68.34%, Let's try to add more layers to the model and see if the test accuracy has any improve.

2-2 Model 2: More-layers model

Goal

We saw previously that shallow architecture was able to achieve 68.34% accuracy only. So, the idea here is to build a deep neural architecture as opposed to shallow architecture which was not able to learn features of objects accurately.
We build Model 2-1 base on reference [4] Model 2-2 add layers base on Model 2-1, Model 2-3 add layers base on Model 2-2. We want to compare the test accuracy of the three models and display the results.
We will build the convolutional neural network with batch size 128 and 200 epochs. the network model will output the possibilities of 10 different categories (num_classes) can belong to the image.

Architecture

Model 2-1	Model 2-2	Model 2-3
Con3-32 Activation Maxpool Dropout ↓ Con3-64 Activation Maxpool Dropout ↓ Con3-128 Activation Maxpool Dropout ↓ Flatten Dense Activation Dense softmax	Con3-32 Activation Maxpool Dropout ↓ Con3-64 Activation Maxpool Dropout ↓ Con3-128 Activation Maxpool Dropout ↓ Con3-256 Activation Maxpool Dropout ↓ Flatten Dense Activation Dense	Con3-32 Activation Maxpool Dropout ↓ Con3-64 Activation Maxpool Dropout ↓ Con3-128 Activation Maxpool Dropout ↓ Con3-256 Activation Maxpool Dropout ↓ Con3-512 Activation Maxpool Dropout

	softmax	↓ Flatten Dense Activation Dense softmax

Table 6: Compare of architecture of Model 2

- **Activation layer:** Some of activation layer functions which are used in CNN are: Identity, tanh, Arc Tan, Relu (rectified linear unit). Inputs are fed to these functions and they give corresponding outputs. The purpose of activation layers is to introduce non-linearity, which will make the training faster and more accurate.
- **Dropout** is a regularization technique for neural network models proposed by Srivastava, et al. in their 2014 paper Dropout: A Simple Way to Prevent Neural Networks from Overfitting. Dropout is a technique where randomly selected neurons are ignored during training.
- **Softmax layer**: Softmax is implemented through a neural network layer just before the output layer. The Softmax layer must have the same number of nodes as the output layer.

Output the probability that the sample belongs to each class.
We use the Stochastic Gradient Descent algorithm to optimize the weights on the backpropagation. Set the momentum parameter as 0.9, and the others as default.

Results

1. Parameter setting and result comparison

	Model 2-1	Model 2-2	Model 2-3
batch size	128	128	128
epoch	200	200	200
test loss	0.5701	0.5877	0.5346
test accuracy	0.8088	0.8089	0.8315

Table 7: Results summary of Model 2

In accordance with the performance of test accuracy, we choose Model 2-3 with the accuracy 83.15%.

2. Tran/Test Accuracy and Loss

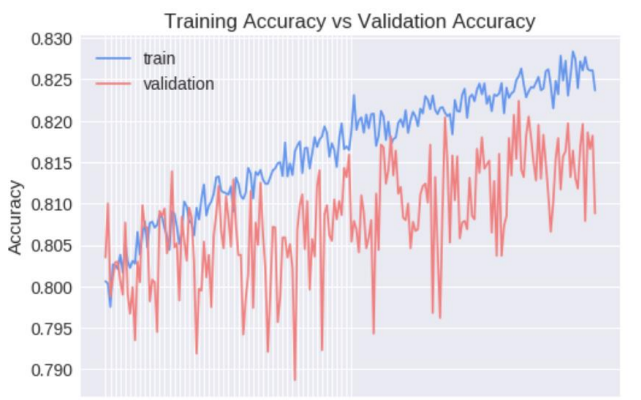
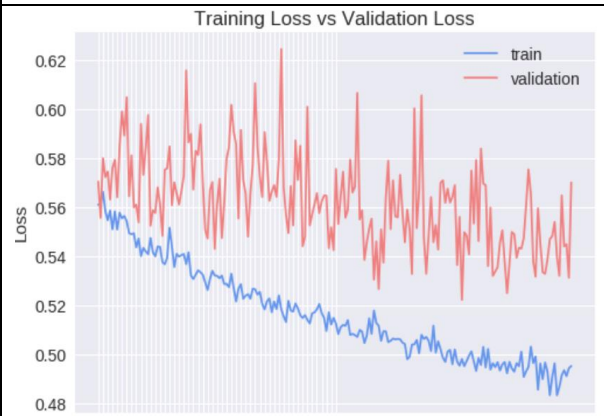
Accuracy	Loss
	

Table 8: Model 2-1 batch size 128 epoch 200

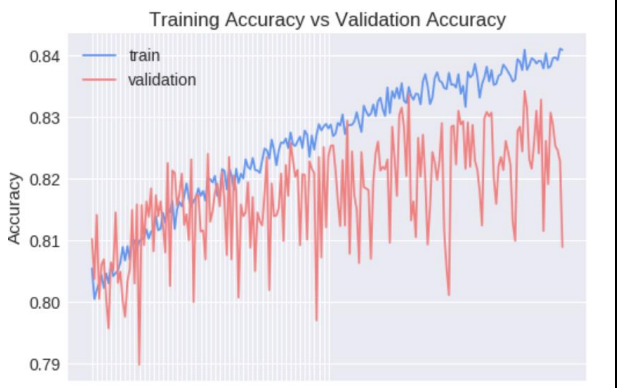
Accuracy	Loss
	

Table 9: Model 2-2 batch size 128 epoch 200

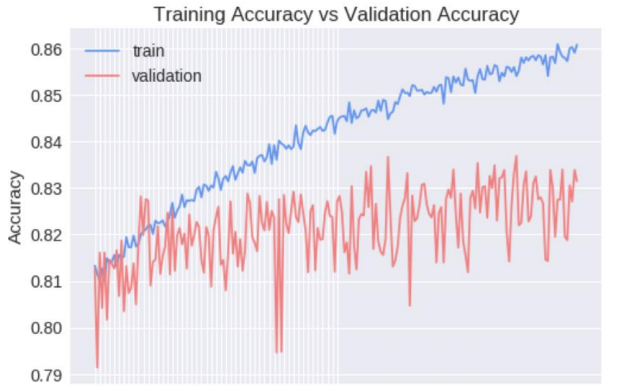
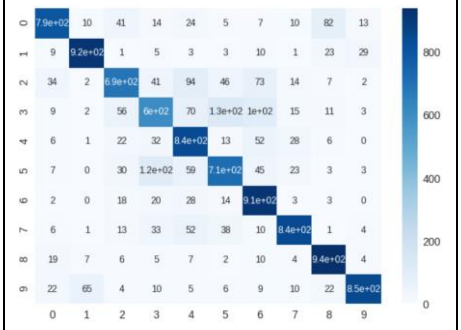
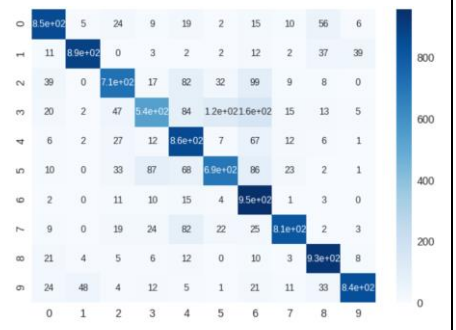
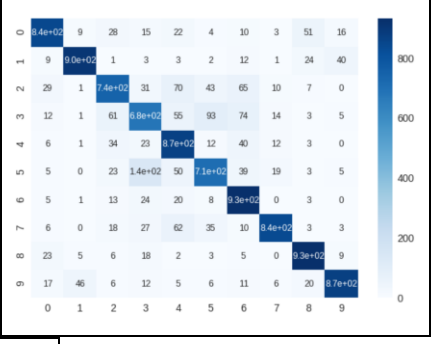
Accuracy	Loss
	

Table 10: Model 2-3 batch size 128 epoch 200

Notice that the test result is 83.15%, which performance better than the previous simple model.
We try to prevent overfitting by adding dropout layers.

3. Confusion Matrix

We use confusion matrix to see which of the classes are more likely to be misclassified.

Model 2-1	Model 2-2	Model 2-3
		

0	1	2	3	4	5	6	7	8	9
airplane	automobile	bird	cat	deer	dog	frog	horse	ship	truck

Table 11: Confusion matrix result of Model 2

In Model 2-3 the darkest is the sixth and eighth, which means frog and ship are less likely to be misclassified, whereas 32% of cats are given the wrong labels. 9% of cats are misclassified as dogs.

4. Examples of Mis-classified Images

Figure 5: Right classified and Mis-classified images in Model 2-3

Above all are 32 examples of images that are predicted wrong by Model 2-3 CNN classifiers. The third one in the last row is easy to identity it's a bird but mis-classified as an airplane by the CNN classifiers.

Insights:

The test result is 83.15%, Let's try to do some data augmentation to the model and see if the test accuracy has any improve.

2-3 Model 3: Data augmentation

Goal

We can increase the number of images in the dataset with the help of a method in Keras library named “ImageDataGenerator” by augmenting the images with horizontal/vertical flipping, rescaling, rotating and whitening etc.

We got this idea of data preprocessing from reference [5]. First, set up the deep network architecture. Then doing data augmentation and regularizatiojn.

Architecture

Model 3-1	Model 3-2	Model 3-3
Con3-32 Activation BatchNormalization Con3-32 Activation BatchNormalization Maxpool Dropout ↓ Con3-64 Activation BatchNormalization Con3-64 Activation BatchNormalization Maxpool Dropout ↓ Con3-128 Activation BatchNormalization Con3-128 Activation BatchNormalization Maxpool Dropout ↓	Con3-32 Activation BatchNormalization Con3-32 Activation BatchNormalization Maxpool Dropout ↓ Con3-64 Activation BatchNormalization Con3-64 Activation BatchNormalization Maxpool Dropout ↓ Con3-128 Activation BatchNormalization Con3-128 Activation BatchNormalization Maxpool Dropout ↓	Con3-32 Activation BatchNormalization Con3-32 Activation BatchNormalization Maxpool Dropout ↓ Con3-64 Activation BatchNormalization Con3-64 Activation BatchNormalization Maxpool Dropout ↓ Con3-128 Activation BatchNormalization Con3-128 Activation BatchNormalization Maxpool Dropout ↓

Flatten softmax	Con3-256 Activation BatchNormalization Con3-256 Activation BatchNormalization Maxpool Dropout ↓ Flatten softmax	Con3-256 Activation BatchNormalization Con3-256 Activation BatchNormalization Maxpool Dropout ↓ Con3-512 Activation BatchNormalization Con3-512 Activation BatchNormalization Maxpool Dropout ↓ Flatten softmax

Table 12: Compare architecture of Model 3

- **BatchNormalization**: Batch Normalization is a technique to provide any layer in a Neural Network with inputs that are zero mean/unit variance[6], it is a method to reduce internal covariate shift in neural networks.

We use RMSprop as our optimizer. The RMSprop optimizer is similar to the gradient descent algorithm with momentum. The RMSprop optimizer restricts the oscillations in the vertical direction. Set learning rate 0.001 and decay rate 1e-6.
(opt_rms = keras.optimizers.rmsprop(lr=0.001,decay=1e-6))

Data augmentation:

1. We randomly rotate images within range 15.
2. We randomly horizontal shifts and vertical shifts images within ratio 10%.
3. The images will be randomly horizontally flipped.

Results

1. Parameter setting and result comparison

	Model 3-1	Model 3-2	Model 3-3
batch size	128	128	128
epoch	100	100	100
test loss	0.4830	0.4873	0.5127
test accuracy	0.8755	0.8938	0.8900

Table 13: Results summary of Model 3

Model 3-2 perform better, choose model 3-2.

2. Tran/Test Accuracy and Loss

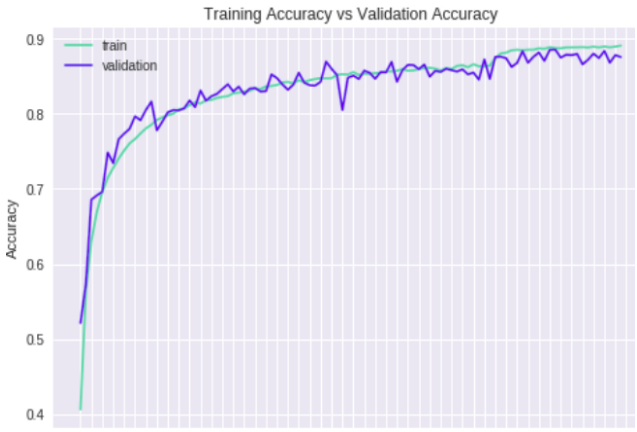
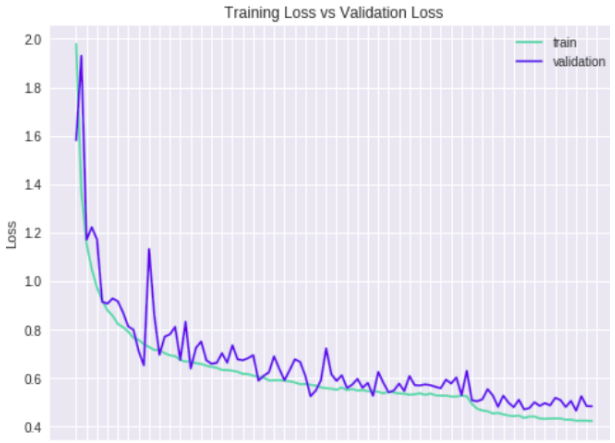
Accuracy	Loss
	

Table 14: Model 3-1 batch size 128 epoch 100

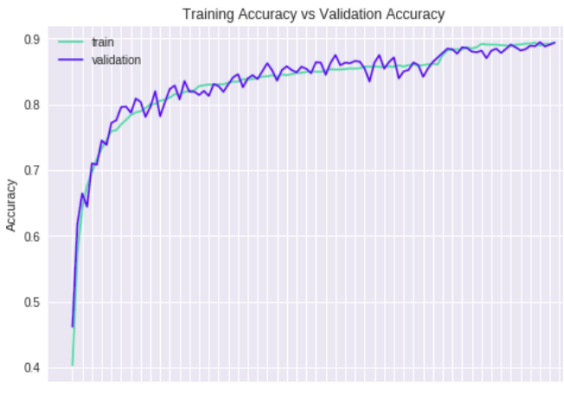
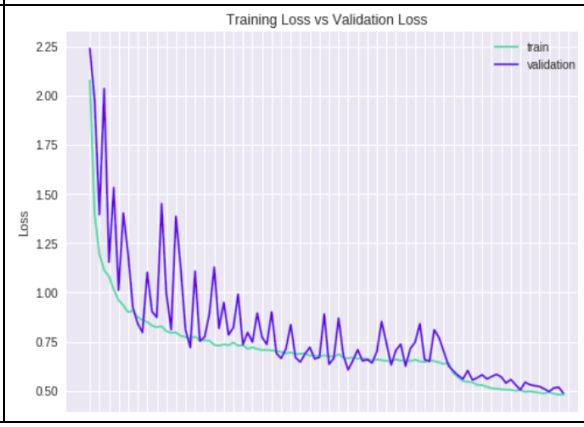
Accuracy	Loss
	

Table 15: Model 3-2 batch size 128 epoch 100

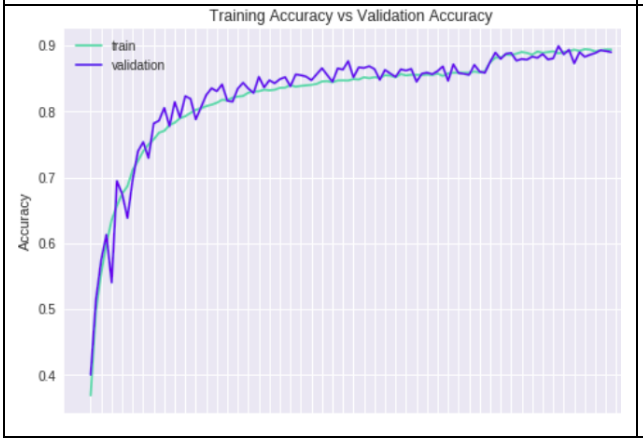
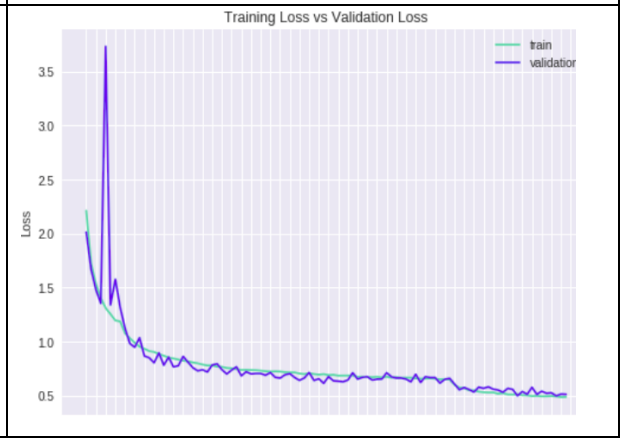
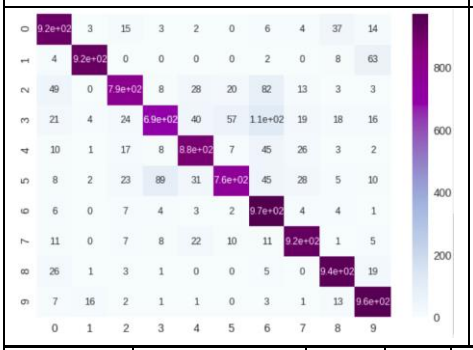
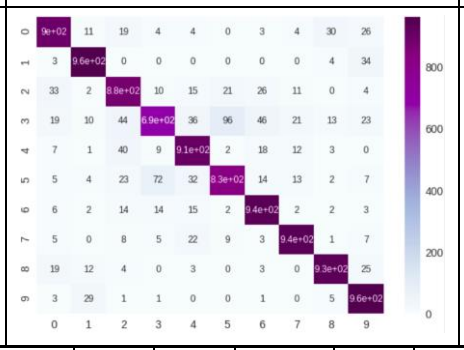
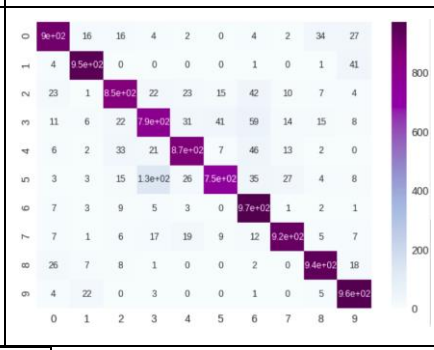
Accuracy	Loss
	

Table 16: Model 3-3 batch size 128 epoch 100

Model 3-3 increase test accuracy from 83.15% to 89.00%. However, the model seems overfitting. So, we choose model 3-2, which perform better with test accuracy 89.38%

3.Confusion Matrix

Model 3-1	Model 3-2	Model 3-3
		

0	1	2	3	4	5	6	7	8	9
airplane	automobile	bird	cat	deer	dog	frog	horse	ship	truck

Table 17: Confusion matrix result of Model 3

In Model 3-2 the darkest is the first and ninth, which means automobile and truck are less likely to be misclassified, whereas 31% of cats are given the wrong labels. 9.6% of cats are misclassified as dogs.

3. Discussion

3-1. Summarize the findings

- Add suitable more layers can achieve higher accuracy.
- Add dropout layer and Batchnormalization can prevent overfitting to some extent.
- Data augmentation is very useful to improve the performance of model.

3-2. Model Comparison

The advantage of multiple layers is that they can learn features at various levels of abstraction. For instance, the first layer will train itself to recognize very basic things like edges, the second layer will train itself to recognize collections of edges such as shapes, the third layer will train itself to recognize collections of shapes like faces and the fourth layer will learn higher-order features like dog or cat. Multiple layers are much better at generalizing because they learn all the intermediate features between the raw input and the high-level classification. At the same time, we should be careful to prevent over-fitting.
Above all, the best performance is model 3-2 with data augmentation, achieve test accuracy 89.38%. We believe that if continuing to train this CNN model with more epochs, the test accurarcy will keep improving.

3-3. Other trials

Wide Residual Network

Deep residual networks were shown to be able to scale up to thousands of layers and still have improving performance but very slow to train[8]. The wide residual networks (WRNs) with only 16 layers can significantly outperform 1000-layer deep networks Also, wide residual networks are several times faster to train[9].
We want to implementation of Wide Residual Networks from the paper Wide Residual Networks in Keras. According to the paper, one should be able to achieve accuracy of 96% for CIFAR10 data set[7].
The WRN-16-8 model has been tested on the CIFAR 10 dataset. It achieves a score of 86.17% after 100 epochs. Training was done by using the Adam optimizer.

1. Parameter setting

Batch size:100
Epoch: 100

Training was done by using the Adam optimizer.

2. Tran/Test Accuracy and Loss

Test loss: 0.6565566521644592

Test accuracy: 0.8617

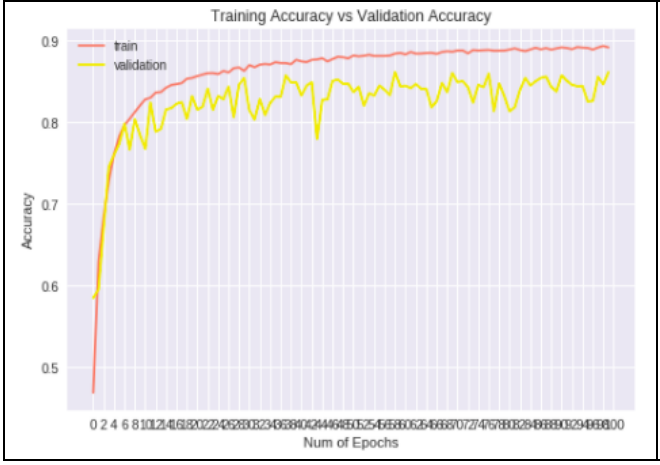
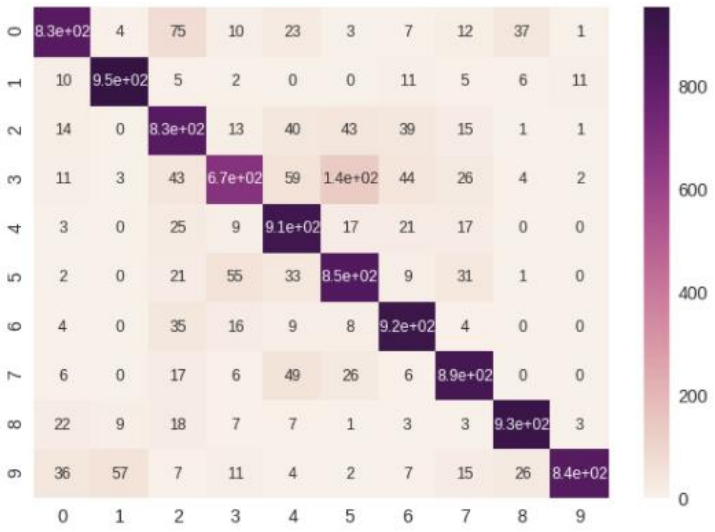
Accuracy	Loss
	

Table 18: Model WRN batch size 128 epoch 100

We don't achieve accuracy of 96% as the paper said, only 86.17%, however the score may improve with further training.

3. Confusion Matrix



0	1	2	3	4	5	6	7	8	9
airplane	automobile	bird	cat	deer	dog	frog	horse	ship	truck

Table 19: Confusion matrix result of Model WRN

In Model WRN the darkest is the No.1, which means automobile is less likely to be misclassified, whereas 33% of cats are given the wrong labels. 10% of cats are misclassified as dogs.

3-4. Future Improvements

- Train the model with more time.
- The WRN-28-10 model could not be used due to GPU memory constraints, which can achieve score of 95.83 %, we want to find a way to implement WRN-28-10 model[10].
- Changing optimizer. Stochastic Gradient Descent algorithm to optimize the weights is probably not the most appropriate algorithm for this dataset.
- Changing the architecture. We need to try neural networks like ResNet and VGGNet or we need to change the activation functions of the layers in the model. Then, we will have a chance to improve the performance.

References

[1] Plotka, S. (2018). *Cifar-10 Classification using Keras Tutorial - PLON*. [online] PLON. Available at: https://blog.plon.io/tutorials/cifar-10-classification-using-keras-tutorial/ [Accessed 16 Dec. 2018].

[2] Keras.io. (2018). *About Keras models - Keras Documentation*. [online] Available at: https://keras.io/models/about-keras-models/ [Accessed 16 Dec. 2018].

[3] Moncada, S. and Moncada, S. (2018). *Convolutional Neural Networks (CNN): Step 2 - Max Pooling*. [online] SuperDataScience - Big Data | Analytics Careers | Mentors | Success. Available at: https://www.superdatascience.com/convolutional-neural-networks-cnn-step-2-max-pooling/ [Accessed 16 Dec. 2018].

[4] Kingma, D. and Ba, J. (2018). *Adam: A Method for Stochastic Optimization*. [online] Arxiv.org. Available at: https://arxiv.org/abs/1412.6980 [Accessed 16 Dec. 2018].

[5] Medium. (2018). *[Deep Learning Lab] Episode-2: CIFAR-10 – Deep Learning Turkey – Medium*. [online] Available at: https://medium.com/deep-learning-turkey/deep-learning-lab-episode-2-cifar-10-631aea84f11e [Accessed 16 Dec. 2018].

[6] Kumar, V. (2018). *Achieving 90% accuracy in Object Recognition Task on CIFAR-10 Dataset with Keras: Convolutional Neural Networks*. [online] Machine Learning in Action. Available at: https://appliedmachinelearning.blog/2018/03/24/achieving-90-accuracy-in-object-recognition-task-on-cifar-10-dataset-with-keras-convolutional-neural-networks/ [Accessed 16 Dec. 2018].

[7] Ioffe, S. and Szegedy, C. (2018). *Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift*. [online] Arxiv.org. Available at: https://arxiv.org/abs/1502.03167 [Accessed 16 Dec. 2018].

[8] GitHub. (2018). *szagoruyko/wide-residual-networks*. [online] Available at: https://github.com/szagoruyko/wide-residual-networks/tree/master/pytorch [Accessed 16 Dec. 2018].

[9] GitHub. (2018). *titu1994/Wide-Residual-Networks*. [online] Available at: https://github.com/titu1994/Wide-Residual-Networks [Accessed 16 Dec. 2018].

[10] Zagoruyko, S. and Komodakis, N. (2016). *Wide Residual Networks*. [online] Arxiv.org. Available at: https://arxiv.org/pdf/1605.07146.pdf [Accessed 16 Dec. 2018].