**REPORT Kaggle: CIFAR-10 Machine Learning.**

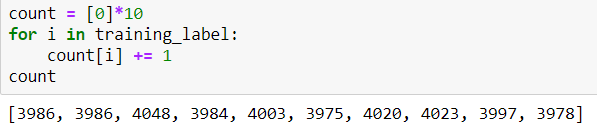
|  |  |
| --- | --- |
| Dataset | <https://www.cs.toronto.edu/~kriz/cifar.html> |
| Language | Python |
| Library | Keras, Numpy, Pandas, Matplotlib |
| Machine Learning Technique | Deep Convolutional Neural Network (ResNet-20) |

**Data**

I have decided to split a portion of the training dataset, 4:1 (80% training, 20% validation) as our validation set. Validation is created such that the final model does not over fit.

Selecting the last file data\_batch\_5 for our validation data.

Selecting the last file data\_batch\_5 does not imply that the training data is not biased. Data could be arranged by its label, so counting is performed.



As we can see the counting shows data a mostly identical in size. This also implies that testing data is balanced in count, because the total number of each label is the same. So we can continue to the next phase.

Besides that we normalize our color channels, because the original channels is integer (0,255). We turn these into floats so the loss will not favor larger values.

The labels are in index format, to train our network it needed to be map to the output layer of the neural network, therefore we will use one hot encoding format. Telling that the final output of the final layer are expected to be the particular neuron.

**Data Augmentation**



We used ImageDataGenerator() from keras.processing.image to create augmentation.

The augmentation in brief:

1. Translation(shift): horizontal and vertical each shift between values from [-10%,10%] of width or height
2. Rotation: an upside down rotation.
3. Flipping: only do horizontal flipping - mirror images.

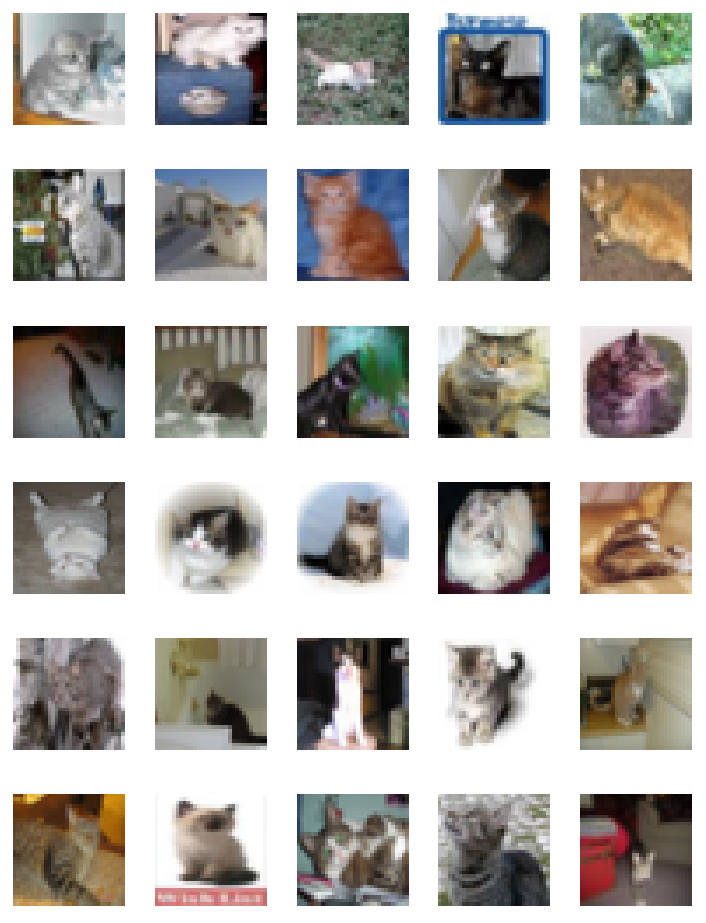
Parameters:

* data\_format: is for the library to know our how our data is placed.
* channels\_last: implies our color channels are placed in the last index, which we always describe our width or height first then our channels RGB at the end.

Data Augmentation is an easy way to quickly obtain more data that is slightly different from the original data. We can possibly include even more complex changes, but it will slow down the overall training progress.

It also prevents our network to overfit our training data. Training data is overfitted when neural network associate particular non-general features from training data. E.g. images of cats could have head on the particular position of the image or the color white cat is more likely to be in the image.

Then high RGB values in the image would cause neurons associate to cats to light up more. If we give our testing with black cats, the model will very likely failed to predict the cats.



*Cat Images in training data*. (mostly white cat.)

Data Augmentation brings away these features, by introducing color shift or positional shift. So that the problem described previously can be prevented.

**Convolutional Neural Network**

Convolutional Neural Network is a form of regularized multilayer perceptron, inspired from monkey’s visual cortexes. We understand that each neurons corresponds to certain feature of the images and position of images shown visually. The features could be patterns like lines, shapes and colours. [(Venkatesan and Li 2017)](https://paperpile.com/c/Yi7PEk/zYdn)

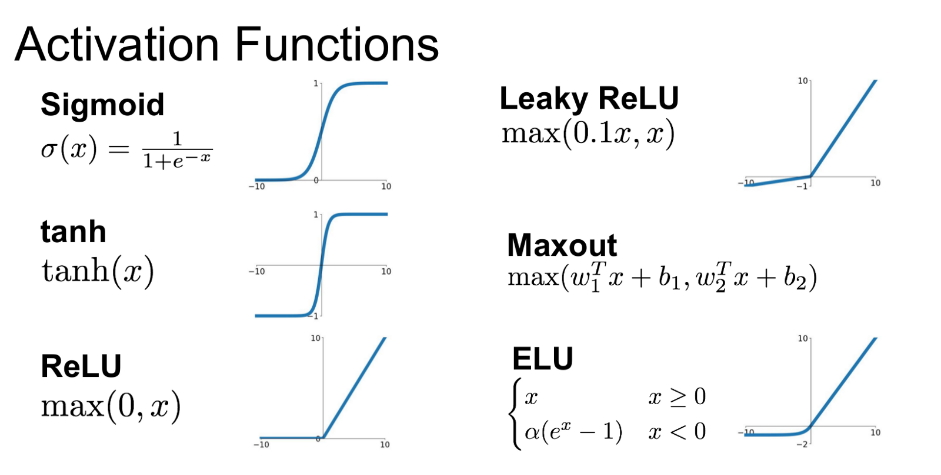
A simple idea is that neural network is formed by multiple layers of tensors. With input data 4 dimensional information (number of images, width, height, and colour channels). The image is then abstracted to another feature map. Continuously until the last layer which usually defined by output labels. The goal is to simulate our visual cortex.

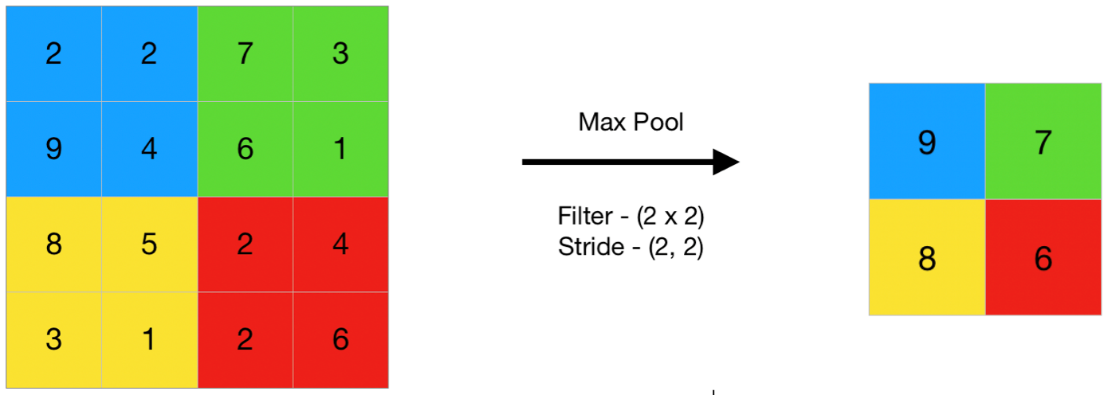
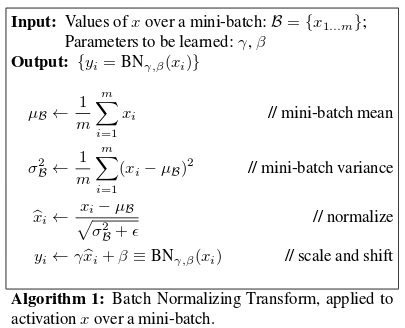
Each layer also reduces the size by pooling the layer previously combining them into a single value.

At the end of feature map, we can connect them to a fully connected dense layer or multiple dense layers. The final layers typically output a vector of real values and the largest can imply the neuron that is lighten up the most, with the probability of being the highest of that label. (Venkatesan, 2017)

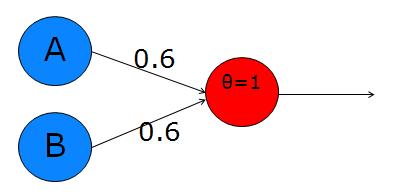
Keras libray have layers where we can learn more about neural networks:

1. Conv2d: Convolutional Layers, each layers contains weights and biases, which is tuned during training process. Numbers from previous layers multiply by the weight and added by the biases to give us the value for the next layers. This value for next layer typically goes through an activation function.



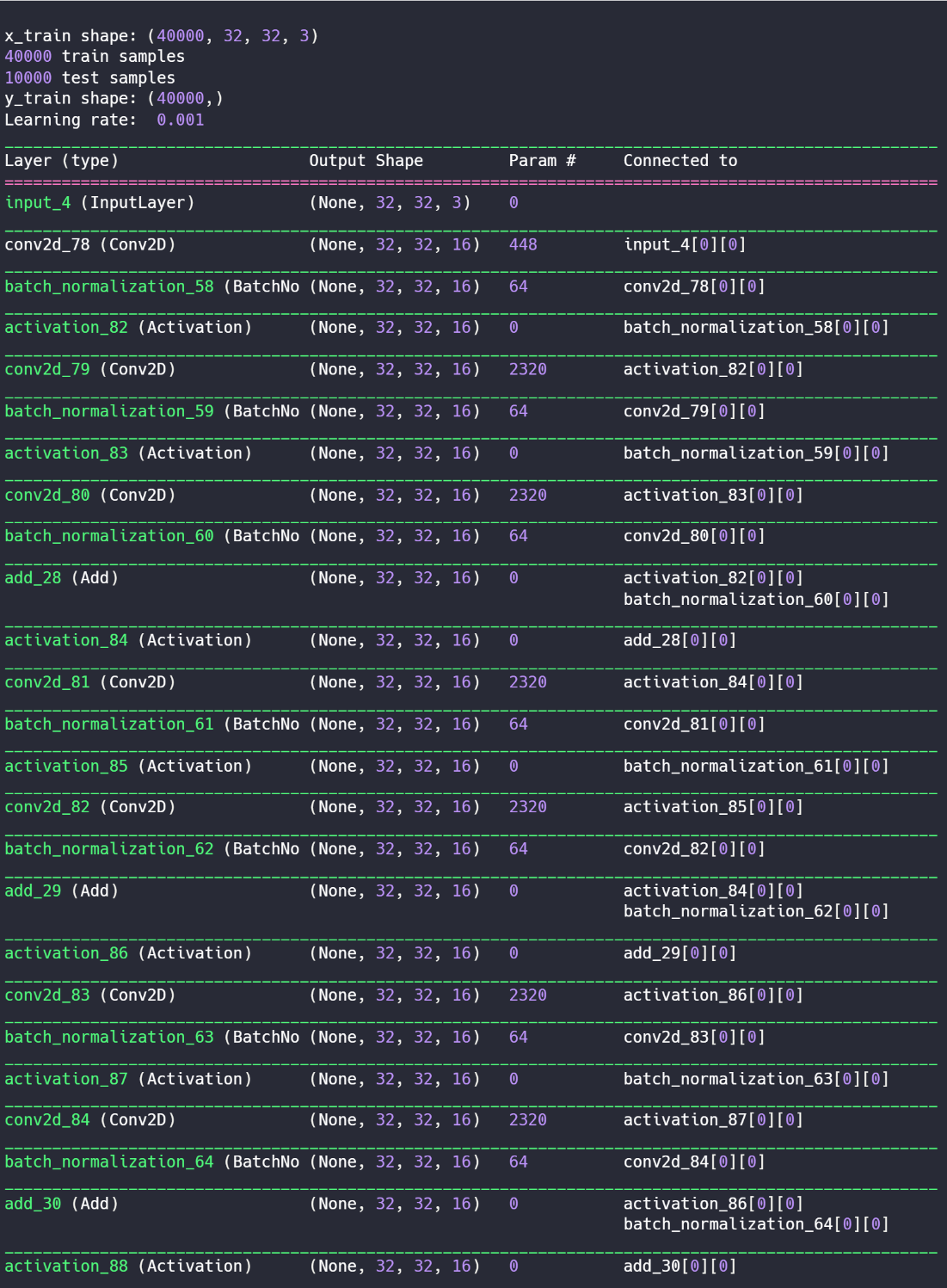
1. Max Pooling: Combining multiple cells of activation into one, by choosing the largest
   1. 
2. Activation: Activation layers activate the values, by mapping it to other values. Some activation functions are hard to calculate, although they give good results. The activation function used in the resnet is (Brownlee, 2019) which maintains a positive mean.
3. Batch Normalization: normalize the layers activation, easy calculation can be made:
   1. 
   2. ( Szegedy & Ioffe, 2015)
4. Flatten: transforming the tensor into a vector
5. Dense: a fully connected single layer perceptron. Which contains bias and weights and a activation function to normalize the values.

Neural networks works in a very interesting way. They could turn different logical gates given a correct weighting and activation function



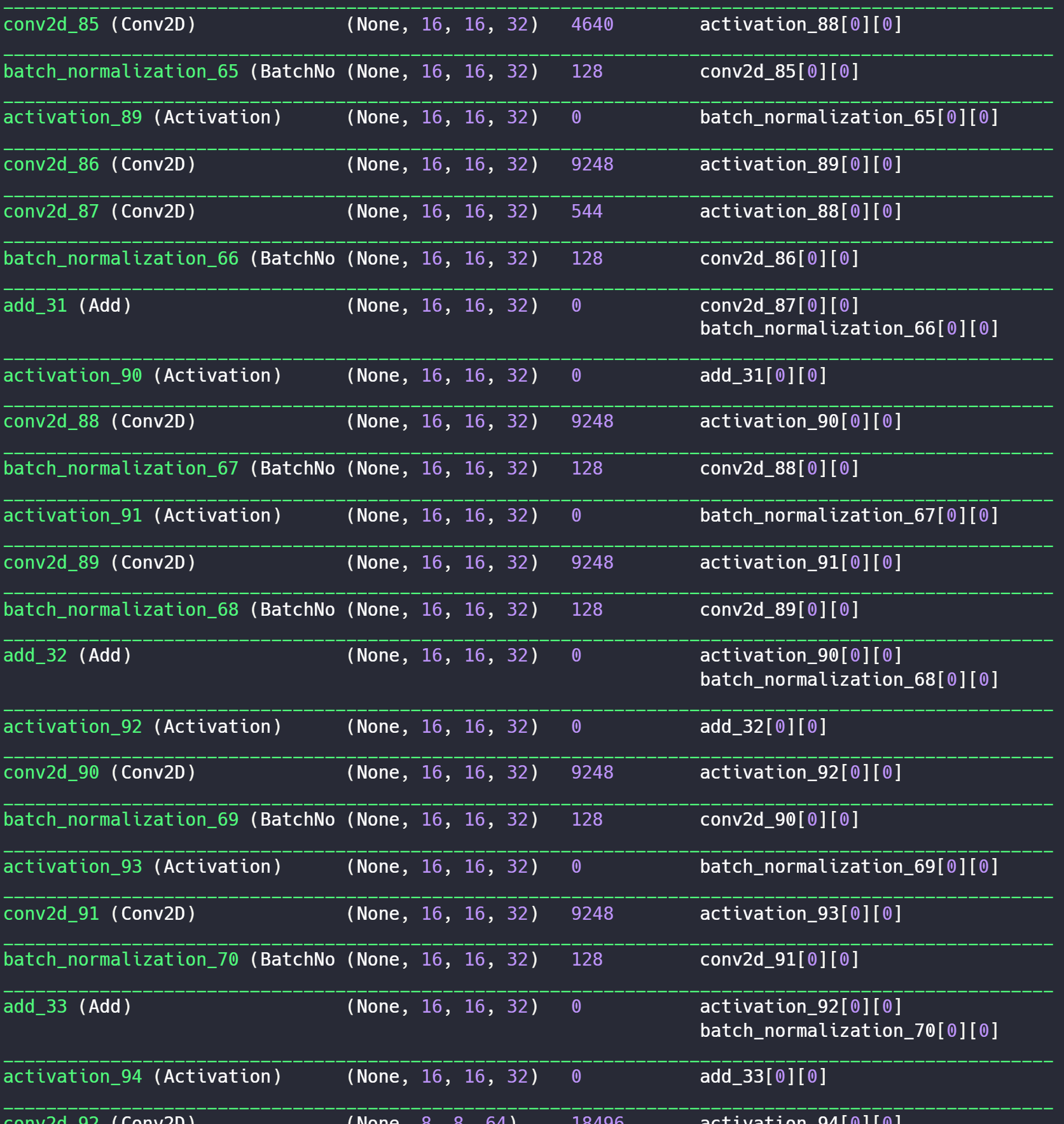
*Example of a neuron acting as an AND gate* (Ben, n.d.)

ResNet aims to ease the training of networks that are deep. Allow increasing accuracy on considerably deeper networks. ResNet are aspected to have lower complexity than their counterparts with the same number of layers. (He, Zhang, Ren, & Sun, 2015)



ResNet Layers

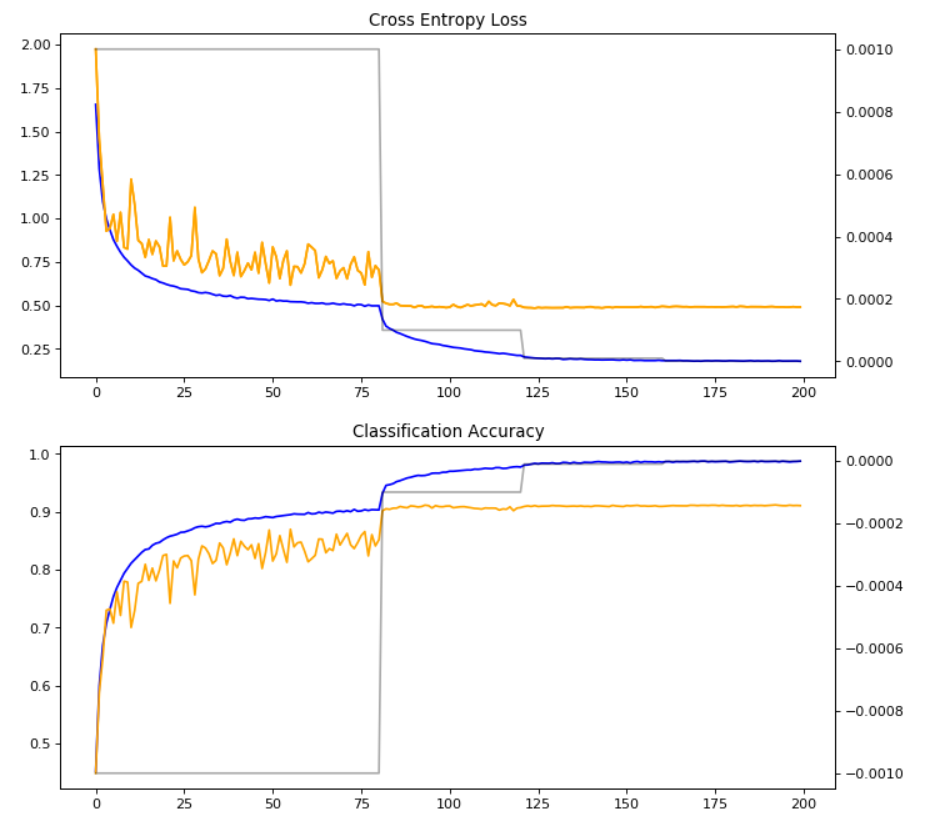
 As our result we can see the input shape is 40000, 32, 32 ,3 we split our data



ResNet Layers

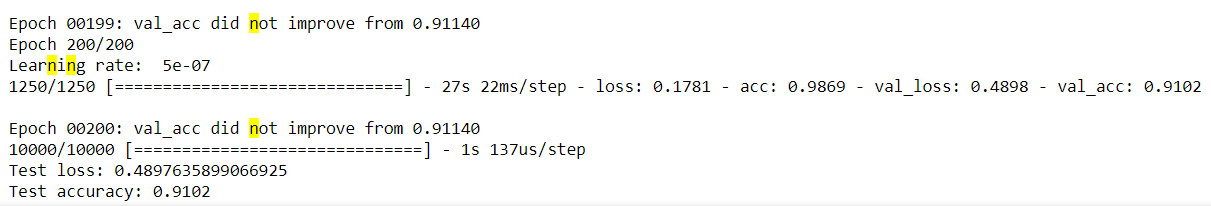


ResNet Layers



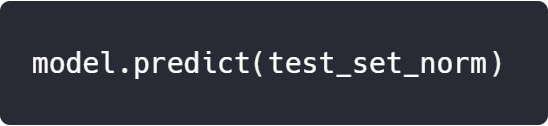
*Cross Entropy loss and classification accuracy*

We reduces our learning rate originally from 1x10-3 to 1x10-4 at 80 epoch,1x10-5 at epoch 120, 1X10-6 at epoch 160 and 5x10-7 180 epochs. We ran our model at 200 epoch and get the final accuracy as:



As you can see, when the learning rate reduce the network gains a significantly lower loss and higher accuracy, we know that the gradient descent are able to reach a lower levels as the step of moving downwards become smaller. This boost of learning knowledge is very interesting as it maps the feature to a mathematical loss function, for which sees that the network learns in a logical pathway.

Finally we input our model and use:



Next the result is an array of floats for each pictures, therefore, we do a reverse action of one hot encoding by selecting the largest value in the result.

From comparison between test result and validation result, I suspect that the dataset does not have **enough data**. Which causes the test result to be **less optimal**. Further updates will be done on Kaggle for personal purposes, the official reports will end here.

My final score on Kaggle: **0.83275** with ResNet-20

**https://lh5.googleusercontent.com/0umIl9GfREU9OLEZMHtktpanSrUMFYAGnPd0hxZPsO4SXIrYfxJOelIMqMNej003uh4i7ySaxQXYD1YwZck_21a4gn2rfBYJuXqtz4dC_w_XrQfZ6yodcpr8Tq6DHVSrNog1QOZG**

*Screenshot from* [*https://www.kaggle.com/c/cs4487cp/submissions*](https://www.kaggle.com/c/cs4487cp/submissions)

Next training phase: ResNet-29 with 50,000 training data.

# Works Cited

Szegedy, C., & Ioffe, S. (2015). *Batch Normalization: Accelerating Deep Network Training b.* Google Research.

Ben. (n.d.). *Computation neuroscience in excel*. Retrieved from Toritris: http://toritris.weebly.com/perceptron-2-logical-operations.html

Brownlee, J. (2019). *A Gentle Introduction to the Rectified Linear Unit (ReLU).* Retrieved from https://machinelearningmastery.com/rectified-linear-activation-function-for-deep-learning-neural-networks/.

He, K., Zhang, X., Ren, S., & Sun, J. (2015). *Deep Residual Learning for Image Recognition.*

Venkatesan, R. a. (2017). “Convolutional Neural Networks.” https://doi.org/10.4324/9781315154282-4. *Convolutional Neural Networks in Visual Computing.*