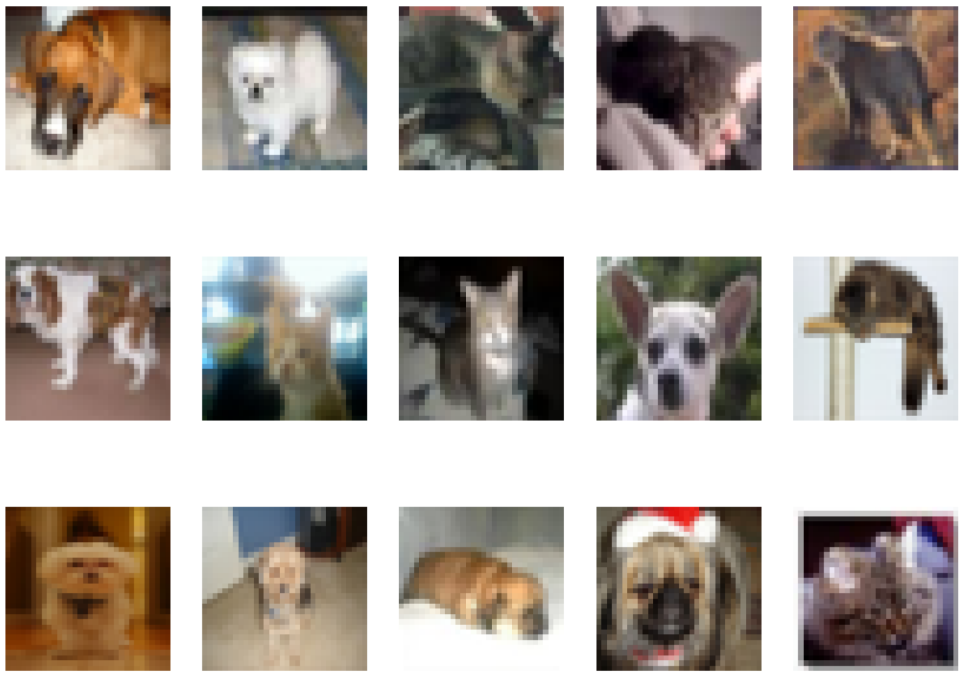
1. The result picture:



2. 5000 cat pictures and 5000 dog pictures in training dataset.

5000 cat pictures and 5000 dog pictures in validation dataset.

1000 cat pictures and 1000 dog pictures in test dataset.

3. We use validation dataset to evaluate our machine learning choices. If we do not improve the validation, it will influence how we evaluate the performance of the training to avoid overfeeding that the learner can only specifically perform well on predicting the training dataset. Thus, we can tweat the parameters and structures for the ML algorithm.

4.

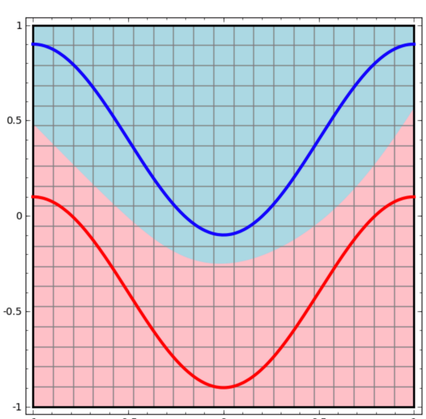
a) (3,5,3): The first 3 means that 3 color-channels for input, 5 means that 5 color-channels for output. The last 3 mean that kernel size is 3*3 which is a convolutional layer. This means that we divide the input image into 3-color-channels, and we convolve the input with 5 3*3 kernels because we would like to 5 channels for output by the convolution.

b) The MaxPool2d is to apply 2D max pooling over an input signal composed of several input planes. We apply this because it can help over-fitting by providing an abstracted form of representation and it can reduce the computational cost by reducing the number of parameters to learn and providing basic translation invariance to the internal representation.

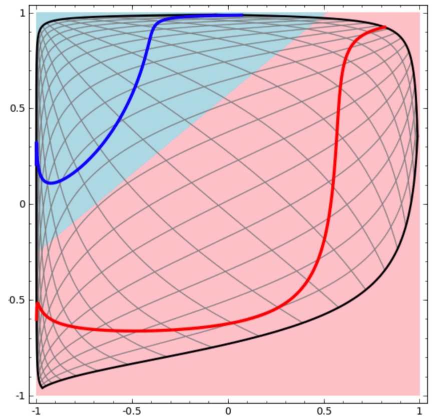
c) I think "fc" represents "fully connected layers", which connect every neuron in one layer to every neuron in another layer.

d) The activation function is "relu"(ReLU), rectified linear units.

5. Two linear layers can increase the accuracy. Two linear layers can increase the capacity of the neuron network since it is activated. Since the model with one linear layer can only predict by linear function, two linear layers can make it predict by non-linear model, which is more complex and more accurate to the real data. Thus, two linear layers is more powerful. Generally, for example, we have first linear layer like this to make decision:



Then, if we add another linear layer to make decision, the output will still be linear. However, the data transformed like this:

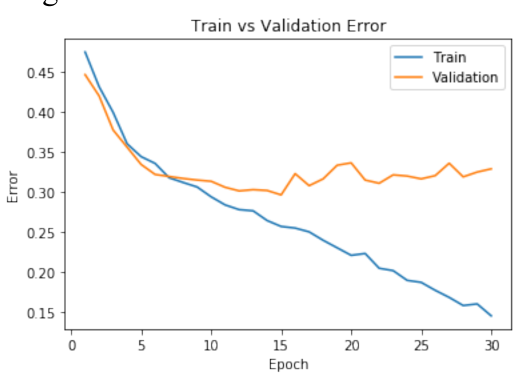


As we can see, it simulates non-linear model which could reduce the noise.

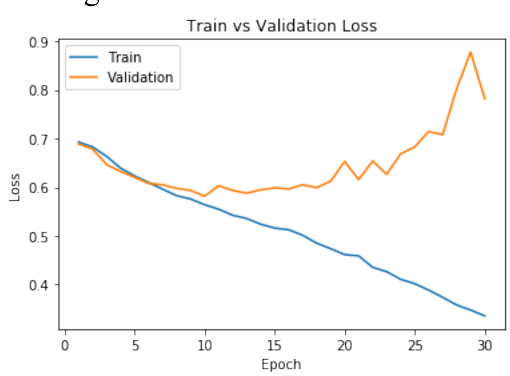
6. Small network error:

Small network loss:

Large network error:



Large network loss:

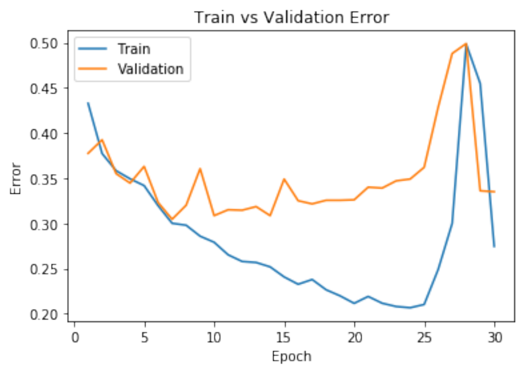


7. Small network is “underfitting” the data because the error value of training data is up and down repeatedly. Thus, our model does not perform well on the training data, so it is “underfitting”.

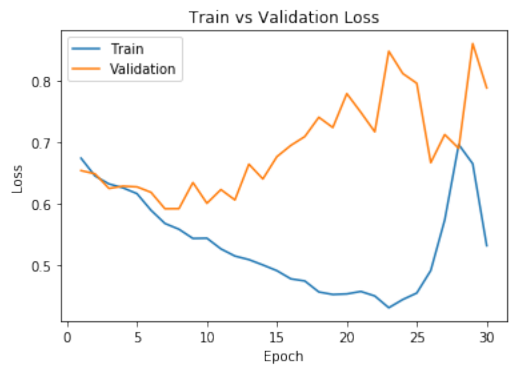
8. Large network is “overfitting” the data because the error value is decline continuously to a very low value for the training data, but the error value for validation data remains about 0.30. It indicates that our model performs well on the training data but does not well on the validation data, so it is “overfitting”.

9. Learning rate = 0.1

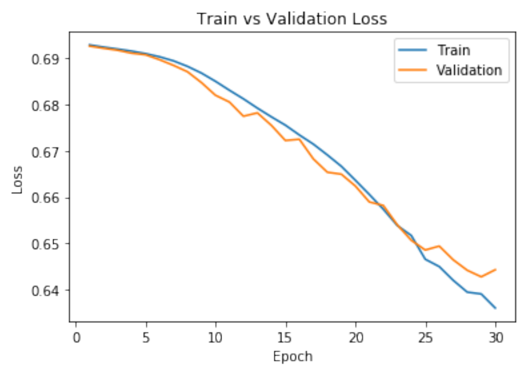
Large network error:



Large network loss:



Learning rate = 0.001



10. According to the graph above, when the learning rate = 0.001, the model has the smallest loss.

11. Code:

```
class DropoutNet(nn.Module):
    def __init__(self):
        super(DropoutNet, self).__init__()
        self.name = "dropout"
        self.conv1 = nn.Conv2d(3, 5, 5)
        self.pool = nn.MaxPool2d(2, 2)
        self.conv2 = nn.Conv2d(5, 10, 5)
        self.dp1 = nn.Dropout()
        self.fc1 = nn.Linear(10 * 5 * 5, 32)
        self.dp2 = nn.Dropout()
        self.fc2 = nn.Linear(32, 1)
```

```
def forward(self, x):
    x = self.pool(F.relu(self.conv1(x)))
    x = self.pool(F.relu(self.conv2(x)))
    x = x.view(-1, 10 * 5 * 5)
    x = self.dp1(x)
    x = F.relu(self.fc1(x))
    x = self.dp2(x)
    x = self.fc2(x)
    x = x.squeeze(1) # Flatten to [batch_size]
    return x
```

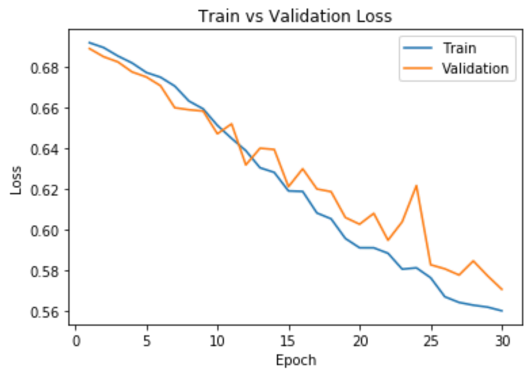
12. Loss graph for dropout model:

13. Dropout layer makes training loss greater than validation loss because of regularization. When dropout layer disables some neurons, then some information about each sample is lost, and the subsequent layers attempt to construct the answers basing on incomplete representations. Thus, the training loss is higher because it makes harder for the network to give the right answers. However, for validation, all of the units are available, so the network has its full computational power - and thus it might perform better than in training.

14. I would like to increase the patch size to 128.

Error graph:

Loss graph:



In a reasonable range, increasing the batch size would increase the model. Thus, I would like to increase the patch size to 128.

15. Since the validation set is to help detect over-fitting and to assist in hyper-parameter search, we can evaluate the performance of the model by trying different parameter and keep the best trained model. The test set allows us to compare different model in an unbiased way based on the data that is not involved in any training process. Thus, we can choose a better model depending on it, which is our final goal.

16. The evaluation function returns that (0.304, 0.5794589556753635). Thus, the error is 0.304, the loss is 0.5794589556753635.

17. The models usually perform better on the validation sets because the hyper-parameters are adjusted based on the validation sets in some way, but the test sets do not involve in any training process. Thus, the model should perform better on validation sets. In some cases, the model

performed better on validation sets because there are meaningful differences between the two data sets. However, in this case, we have much larger validation sets than the test sets, and we may include a variety of cases of cats and dogs in the validation sets which increase the accuracy of the trained models on test sets. Thus, in this case, the models are less likely perform better on the validation set than the test sets.