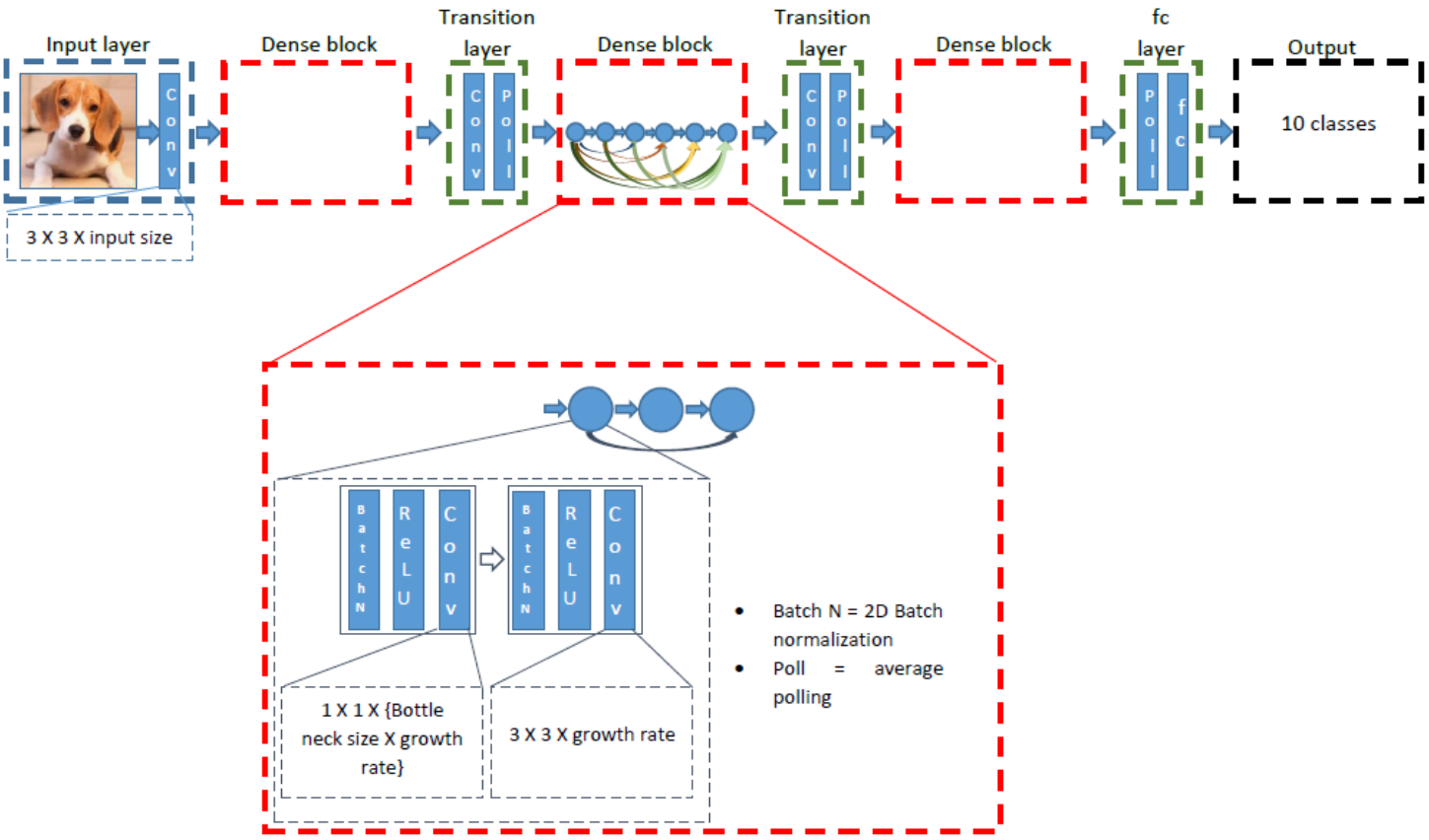
Deep learning HW2

Submitted by: Liron McLey 200307791, Matan Hamra 300280310 & Ariel Weigler 300564267

Model Architecture



The design of the network is based on reference 1. Briefly, we begin with an input layer performing a 3x3 convolution increasing the number of features to the input size. The basic network design is of three dense blocks. In each dense block the output of each dense layer (blue circle in the scheme above) is passed as input to all subsequent dense layers within the block, we used 16 such layers in each block. Each dense layer consists of two basic building blocks, a batch normalization followed by a ReLU non-linearity and a 1X1 convolution that acts as a bottle neck block, followed by another set of BN, ReLU and convolution this time with a 3X3 kernel, as we found this to be the most informative kernel. The networks growth rate and bottle neck size, determine the output size of each dense layer and the dimensions of the bottleneck block in each dense layer respectively. Hence, each dense layer in the block learns only growth rate features while receiving as input all the outputs of previous layers. Between each dense block we introduced a transition layer, consisting of a BN, ReLU and a 1x1 convolution followed by average pooling of 2 with stride 2 to down – sample the image. Before the final fully connected layer, another pooling step is performed, this time with a compression factor determining the number of inputs features to the fully connected. A low compression ratio down

samples the number of features more. By selecting a low growth, e.g. three we were able to construct a network that has less than 50,000 parameters, and yet the connections between layers close to the input and layers close to the output are relatively short making the network extremely effective.

We used SGD as optimizer and CrossEntropyLoss as our criterion.

The following table describes the three **best** hyper-parameter and architecture configurations we ran. We attempted different data augmentation regimes, input layer sizes, number of layers per block, growth rates, bottleneck sizes and compression rates.

	Option A	Option B	Option C
Tot. number of parameters	48870	49136	48870
Input layer size	24	24	24
Growth rate	3	3	3
# of epochs	300	300	300
Learning rate schedule	$\begin{cases} 0.1, & epoch < 150 \\ 0.01, & 150 \leq epoch < 225 \\ 0.001, & 225 \leq epoch < 300 \end{cases}$	$\begin{cases} 0.1, & epoch < 100 \\ 0.01, & 100 \leq epoch < 175 \\ 0.001, & 175 \leq epoch < 250 \\ 0.0001, & 250 \leq epoch < 300 \end{cases}$	Same as option B
Data augmentation	Random crop + padding = 4 + random horizontal flip (p=0.5)	Same as option A	Same as option A + random rotation (45 degrees)
Block configuration	16 x 16 x 16	16 x 16 x 16	16 x 16 x 16
Bottleneck size	4	3	4
compression	0.25	0.48	0.25
Accuracy	**90.65**	**90.75**	**90.24**

As is evident, option B performed best.

Training procedure

The training and test sets downloaded to the computer, shuffled and normalized according to the mean and standard deviation of the CIFAR10 training set. On the train set only, we performed data augmentation regimes prior to normalization. We used the architecture and parameters described above to construct our neural network. We then ran each epoch as follows:

1. A working batch is selected from the training set, over which the code:
    a. Runs forward propagation
    b. Calculates the loss value and error
    c. Uses backward propagation and updates the network weights
2. The updated model is run over the test set
3. The loss and error of the test set is calculated

Summary

Given the constraints of this exercise, we set out looking for a network configuration that is deep, and yet can supply adequate results with a relatively low number of parameters. This led us to the paper, Huang et al. 2018[1], describing the network configuration we presented previously, DenseNet. We used the network from the paper as our basis and played with different architecture configurations to match the constraints of the exercise, namely the extremely low number of parameters. The success of this network attributed to the short connections between the input layer and the rest of the layers, even the deepest ones, achieved by concatenating the outputs of each previous layer as input to the subsequent layers.

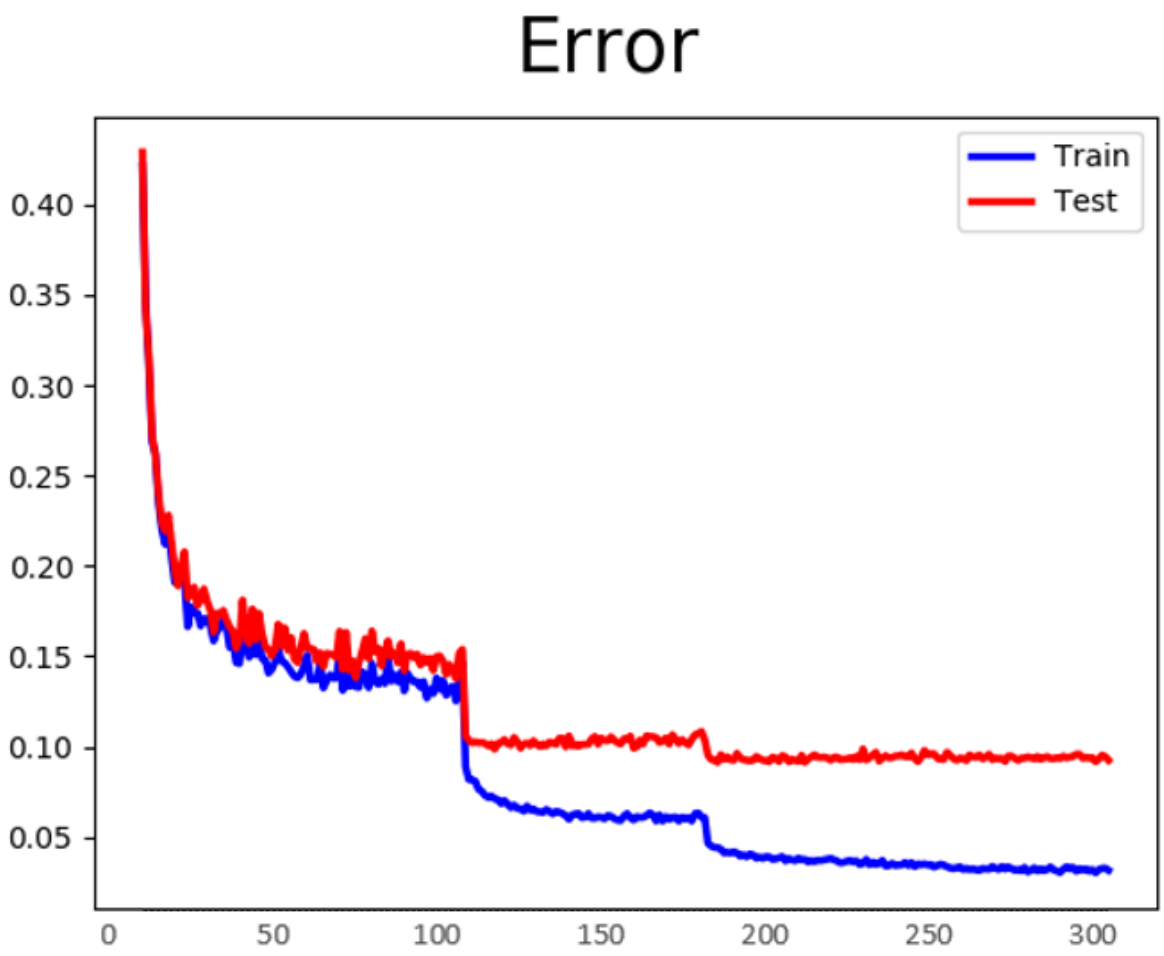
From our experiments, we can conclude the following:

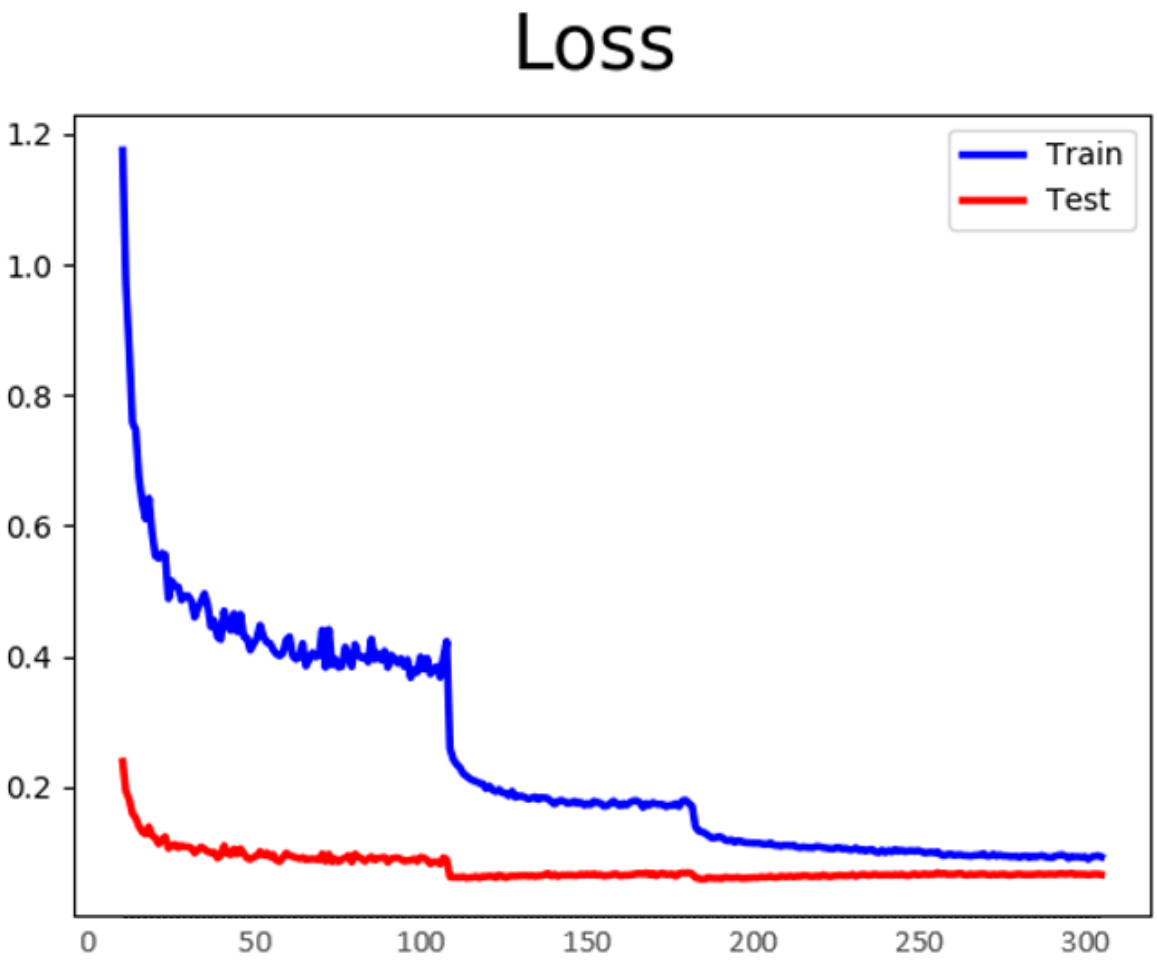
1. **A learning rate scheduler is better than a constant learning rate.** We started with a relatively high learning rate and decreased it every X epochs (see table at the beginning). This gave us much better results than when we ran with a constant value.
2. **The higher the compression, the better:** The run which gave us our best results was also that with the highest compression value. We think the reason for that is that a higher compression value extracts more general features from the images.
3. **Model architecture, deeper is not always better:** the parameters constraint imposes a trade – off between the number of layers in each block and the growth rate**.** As the network depth increases, the growth rate must decrease to keep the same number of parameters**.** Having experimented with thin (low growth rate), deep (over 100 layers) networks and thick, shallow networks we found that the best results are obtained by choosing an intermediate value for both, aka growth rate = 3 and block configuration of (16, 16, 16).
4. **Data augmentation doesn't necessarily help improve results:** At first we thought that applying more data augmentation transforms would help the network learn more general features of the dataset. However, we found that is not necessarily the case, you can see the run where we added random rotation (option C) did worst. We are not sure why that is, after all, this effectively increases our trainable dataset. Perhaps had we let run for more epochs it would have reached better results.

[1] https://arxiv.org/pdf/1608.06993.pdf

Results

Our best results: option B, accuracy = 90.75%




Loss
Train
Test
1.2
1.0
0.8
0.6
0.4
0.2
0
50
100
150
200
250
300