IFT6135 Hiver 2018. **Representation Learning**

Assignment 2 - Practical Part. Convolutional Networks

Professor: Aaron Courville

Amina Madzhun, student id: 20052277
Matthew Scicluna, student id: 20114033
Wenhui Peng, student id: 20087087

March 14, 2018

1. **Regularization : weight decay, early stopping, dropout, domain prior knowledge**

For this question the MNIST dataset was used. We tried in practice, evaluated different regularization techniques in order to improve generalization.
In the first two sections of this problem we trained for 100 epochs the MLP model architectures which are two hidden layers with 800 units using ReLU activations. We used SGD with learning rate of 0.02, minibatch size of 64, and the Glorot Normal for weight initialization.

(a) *Early stopping and weight decay*
We trained two models described above over 100 epochs, one having no regularization and another with L2 regularisation. For L2 regularization we set $\lambda = 2.5$, but rescale the coefficient to adapt the loss for minibatch SGD. We needed to scale lambda proportionally to the subset size, so we took into account the proportion of batch size with respect to the size of training set. We observed the change of all the parameters through L2 norm (the sum of L2 norms of all the parameters of all layers) at each iteration (minibatch update) during the training of both models. The results are presented in Figure 1 on the left. Also we plot the error at the end of each epoch of training, in Figure 1 on the right.

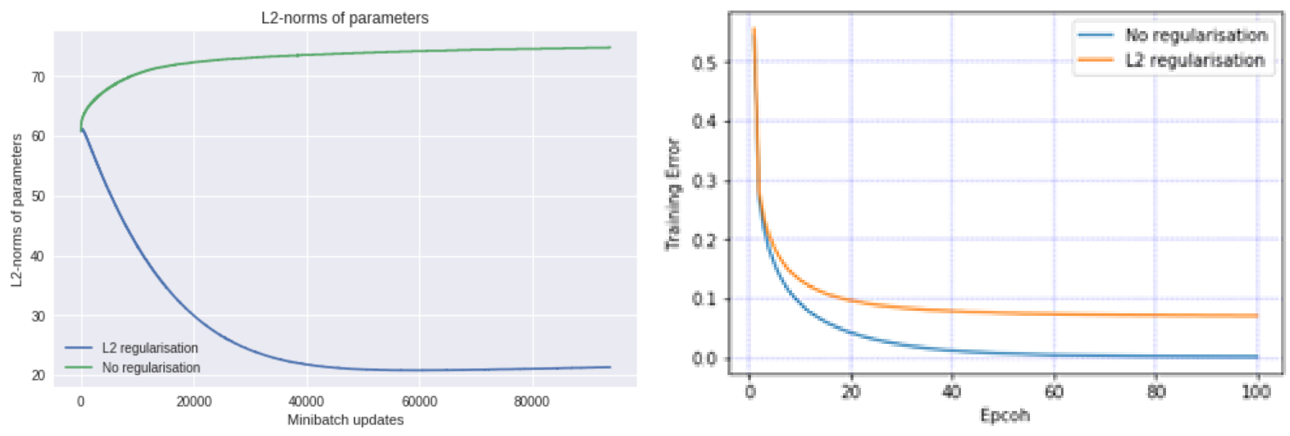
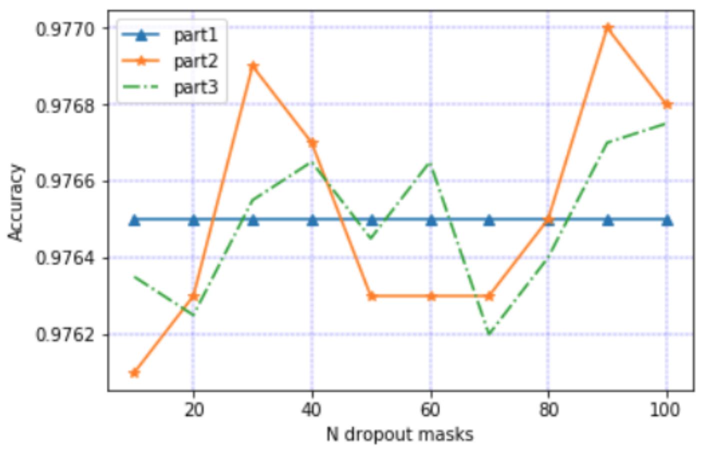


Figure 1: L2 norms of parameters (left). Training error (right)

As we can see, the L2 norm of parameters are decreasing with training epochs in L2 regularization, and increasing with training epochs in no regularization. By adding L2 regularization we get bigger training error as it was expected.
The idea of regularization aims to change the error function to penalize hypothesis complexity. Without regularization, the loss function is empirical risk (training error); by adding the regularization term, we are optimizing the structured risk instead of empirical risk. This is the reason why the value of training error is larger with regularization during training.
The L2 norms of weights are decreasing with training in regularization, and increasing without regularization (increase with model complexity), since L2 regularization adds a squared penalty on the weights: it will cause the magnitude of the weights to be smaller than without penalty.

(b) *Dropout*

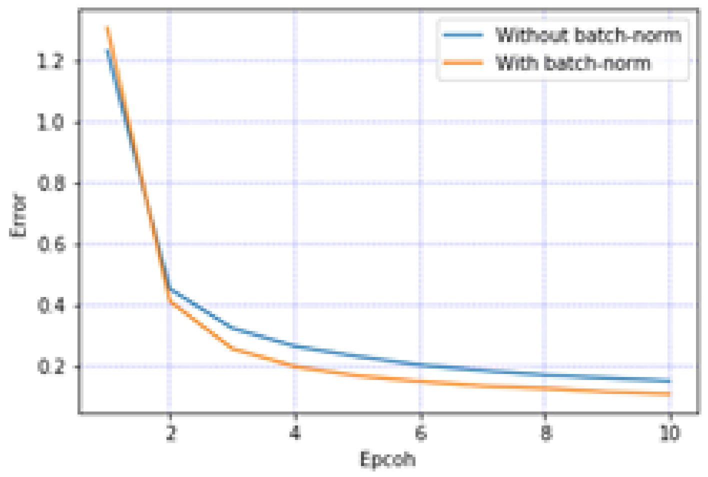


Dropout can be seen as bagging. At each training step in a mini-batch, the dropout procedure creates a different network (by randomly removing some units), conceptually, the whole procedure is akin to using an ensemble of many different networks, each trained with a mini-batch.

We used the same architecture of MLP training over 100 epochs. We applied dropout on the last hidden layer. After training, we had to evaluate it on the test set. The results are presented above. For each data instance, to get a prediction three different schemes were used:

i. multiplication by 0.5 of the hidden layer before prediction
   This scheme performs an efficient approximation of "geometric averaging" over the ensemble of possible subnetworks.
ii. sampling N dropout masks, and averaging the pre-softmax values
   This scheme averages the pre-softmax activations of the ensemble of possible subnetworks
iii. sampling N dropout masks, and averaging the predictions
   This scheme averages the prediction of all the possible subnetworks via the arithmetic mean.

(c) *Convolutional Networks*



Here we used a convolutional network architecture for training on MNIST over 10 epochs, plotting the error at the end of each epoch for two models with and without batch-normalization.
Batch-normalization can improve optimization for faster convergence, as shown in the results: the CNN model learns faster (smaller training error) with batch-normalization than without batch-normalization.

2. **Dogs vs. Cats Classification**
We trained a CNN classifier on the Dogs vs Cats dataset provided by Kaggle. This dataset consisted of color pictures of cats and dogs of various sizes. We processed the images to be of size 64×64 and used 30% of the 20000 training examples as our validation set (while using the provided validation set as our test set). We did not modify the images any further, except to scale the pixels to be between 0 and 1 and then to standardise them to have mean 0 and a standard deviation of 1.

   (a) We trained 2 architectures based on the design from Simonyan and Zisserman (2015) [1]. Each model used the 3×3 convolution size described in the paper, since this size has been shown to be efficient and perform well in practice. The first model (A) had essentially the same architecture as $VGG-A$ in the paper, whereas B and C were scaled down versions. More specifically, for B we scaled the depth of the convolution by a factor of 3.5 in an attempt to account for the difference in sizes of input images (64×64) vs 224×224 in the paper). For C we scaled this even more, but added more layers to try to keep the depth consistent. The architectural details of each network are presented in table 1.

   We trained A using a SGD with a learning rate of $1e-3$, momentum of 0.9, weight decay of $5e-3$ and batch size of 25 for 20 epochs. After the 20th epoch we would continue to train until the validation loss increased, and then doubled the batch size and resumed until we reached a batch size of 200. We chose to increase the batch size rather than decrease the learning rate to speed up the training time. For B and C we used SGD but with a learning rate of $1e-2$ and weight decay of $5e-4$ and batch size of 32. We used early stopping along with a learning rate scheduler instead of increasing the batch size. We used dropout at 0.5 in the fully connected layers. The performance of each network is presented in table 2.

   (b) We plot the training error and validation error for Model C over the first 50 epochs. We chose C since the performance of the 3 models were comparable and it was the simplest to train overall. We tried both vanilla SGD and ADAM, and found that the difference was negligible. We found that batch norm had a significant impact – applying it to each layer increased the accuracy by around 4%.

Table 1: Conv. Net Architectures

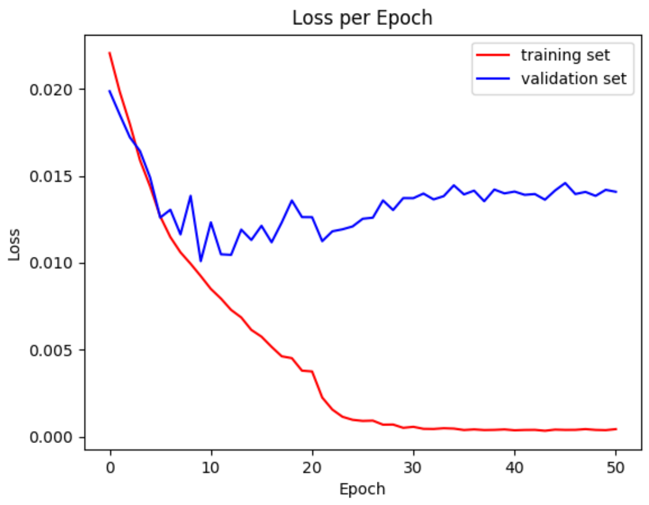
A	B	C
conv $3-64$	conv $3-18$	conv $3-8$
		conv $3-8$
	maxpool	
conv $3-128$	conv $3-36$	conv $3-16$
		conv $3-16$
	maxpool	
conv $3-256$	conv $3-72$	conv $3-32$
conv $3-256$	conv $3-72$	conv $3-32$
conv $3-256$		conv $3-32$
	maxpool	
conv $5-512$	conv $3-144$	conv $3-64$
conv $5-512$	conv $3-144$	conv $3-64$
		conv $3-64$
	maxpool	
conv $5-512$		
conv $5-512$		
maxpool		
FC 2048	FC 2304	FC 1024
FC 2048	FC 2304	FC 512
FC 1000	FC 500	
	Softmax	

Table 2: Model Performance

Model	Train Acc.	Valid Acc.	Test Acc.	Num. Param.
A	99.54%	88.73%	88.02%	$20,270,474$
B	99.89%	87.92%	88.08%	$6,827,526$
C	99.64%	86.86%	86.86%	$4,844,682$

Table 3: Batch size and Model Performance

Batch size	Test Acc.
32	86.86%
50	85.56%
100	85.61%
200	84.56%



(c) We compare the effect of batch size on test error in C. We decided to investigate this due to recent work that suggests that larger batch rates tend to find sharper optima, and these tend to yield results that do not generalise as well as flatter minima. Our results supported this, with our smaller batch size giving the best performance overall. Our empirical experimentation agrees with this view. The results are presented in table 3.

We did a visual analysis of the results. We found the test images which the classifier most confidently miss-classified (using the value of the Softmax output). We present the results in Figures 2. We see that the images in these cases are somewhat deceptive. We did the same for the ambiguous cases (where the classifier predicted around 50% on both classes). The results are displayed in figure 2. Based on inspection, it seemed that the model had the most trouble with figures where there animal was cut-off, or where there were one or more humans and/or other animals in the image. Another common issue was with the resolution – the images had low resolution, so images taken at a distance were difficult to identify simply because the relevant features were totally blurred out. The problem was exacerbated by when the animals were black. Increasing the resolution (and by extension doing random crops on the higher resolution images during training) would probably resolve these issues.

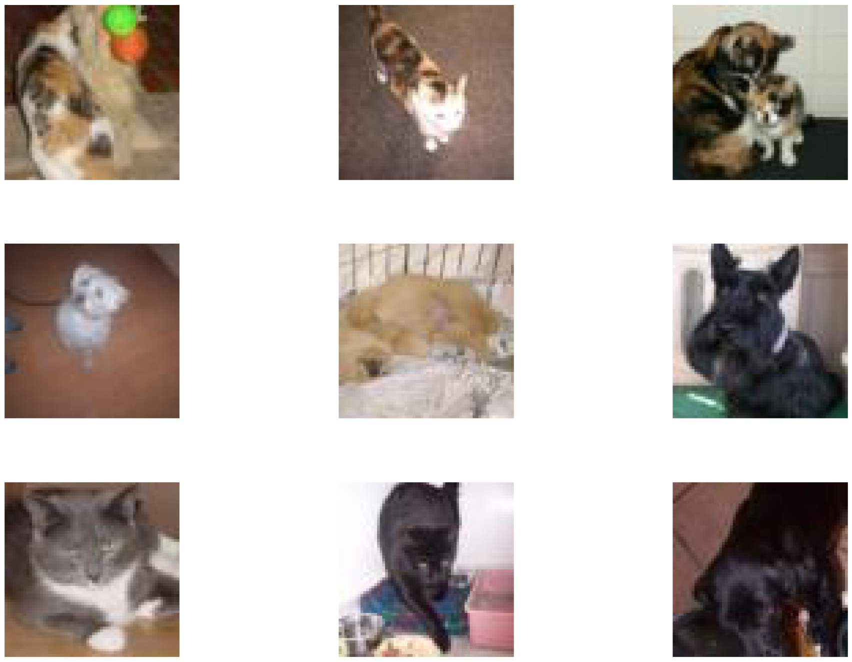


Figure 2: Miss-classifications of the model. First row: cat photos miss-classified as dogs. Second row: dog photos miss-classified as cats. Third row: ambiguous photos (where the output probability was between 49% and 51%).

References

[1] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *CoRR*.