ANIRUDH SETH
aniset@kth.se

DD2424:Assignment 3
Deep Learning in Data Science

2020-04-16

Checking the gradients

The analytical gradients of the weights and the filters for both the hidden layers of the network were compared to the numerical approximations and achieved a mean difference in the range of $\sim$e^{-14} .The results in Table 1 were computed for the first epoch for one batch to validate the gradient computation.The results can be replicated by using seed value of 1 and step size of $h = 1e^{-5}$.

Variable	Mean Difference in Gradient
W	-1.1386e^{-13}
F1	4.8863e^{-14}
F2	-4.4643^{e-14}

Table 1: Mean gradient difference between analytical and numerical approximations.

Efficient Computation

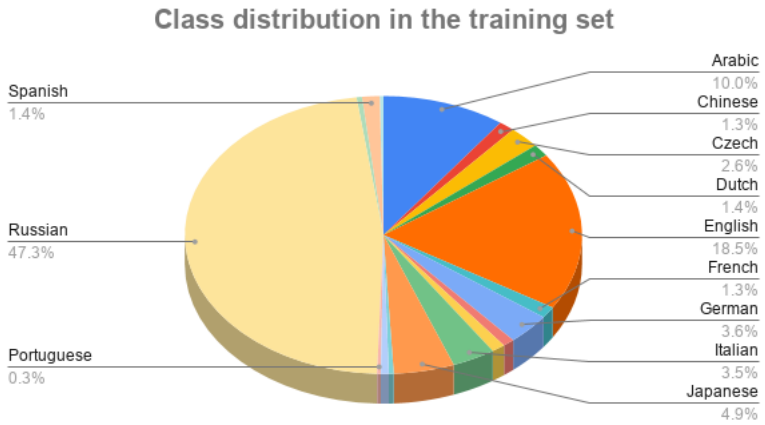
As recommended in Background 5,instead of creating $M^{\text{input}}_{\mathbf{x}_j,k_2,n_2}$, i also implemented a method to create $M^{\text{input}}_{\mathbf{x}_j,k_2\mathbf{j}}$ (the generalization of equation (21) to filters of width k_2 and input of size $n_1 \times n_{1\text{en}_1}$). The efficiency gain was quite significant. The results are summarised in Table 2

Implementation	Total Time	s/it
Slow Method	3:02:43	55.09s/it
Efficient Method	05:37	1.71s/it

Table 2: Computation time improvement using efficient computation.

Compensating for Class Imbalance

The chart below clearly shows a class imbalance in the training set.



Random Re-Sampling: To account for imbalanced data ,as recommended, for each epoch of training I randomly sampled the same number (count of the smallest class) of examples from each class and this became the new training set for this epoch. A model created with imbalanced dataset has a bias towards the majority classes and can result in poor generalization. This is evident from the results in Table 3. With imbalanced data, the model achieves a very high accuracy on the training set 82.11% but has a significantly poor performance on the validation set 36.90%. This clearly shows that the model is over fitting the training data and has a poor generalisation on unseen data. On the other hand the same model (with the same parameters) when trained with balanced data has better accuracy of 46.42% on the same validation set and is no longer over fitting the training data as much.

N_1:20 K_1:5 N_2:20 K_2:3 ρ:0.9 ETA=0.01		
Accuracy	**Unbalanced Dataset**	**Balanced Dataset**
Training Set	0.8211	0.5925
Validation Set	0.3690	0.4642

Table 3: Training results with and without compensating for the unbalanced dataset.

The loss, accuracy and the confusion matrix (as a colormesh) for both models is presented below.

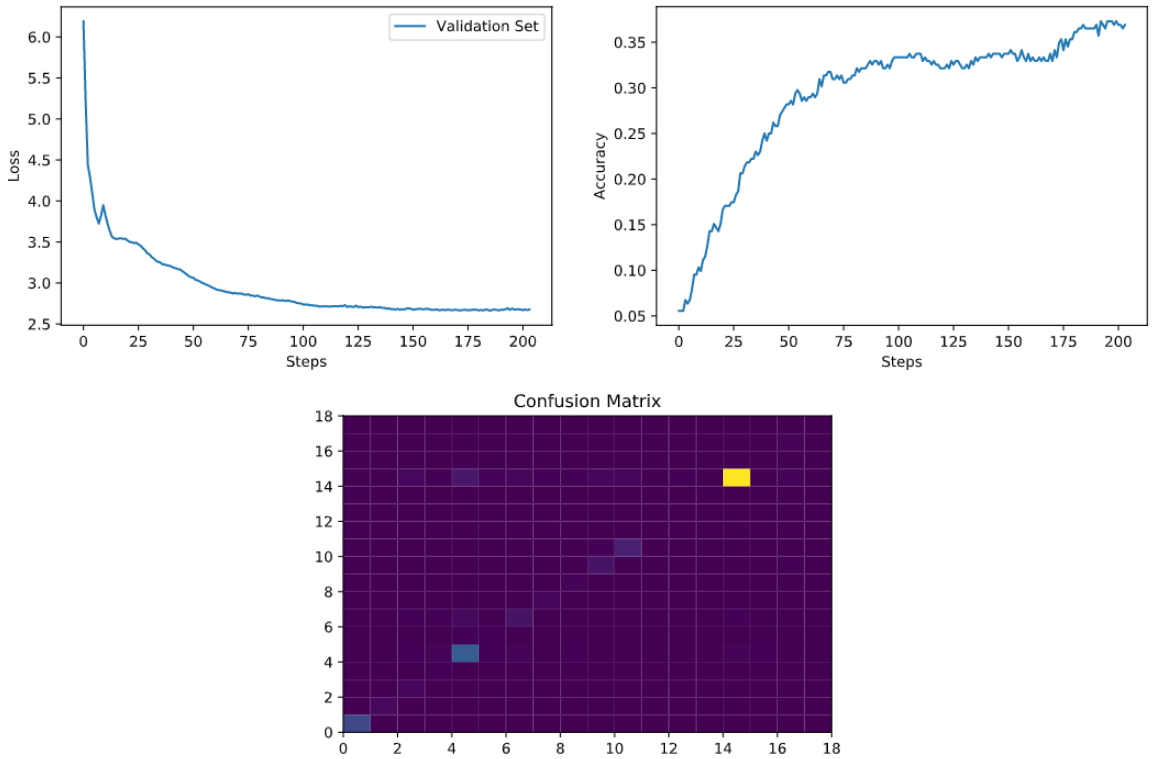


Figure 1: Model trained with unbalanced dataset. Training Loss and Accuracy on the validation set. Confusion Matrix as a color mesh on the bottom.

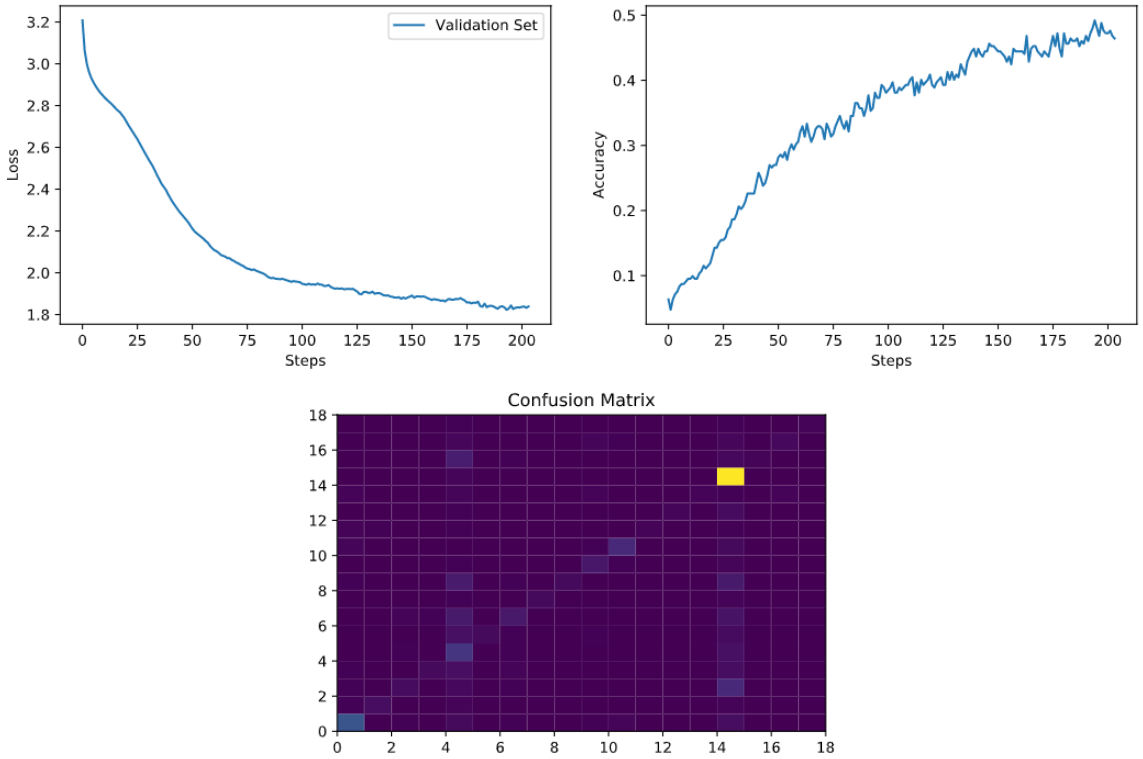


Figure 2: Model trained with balanced dataset. Training Loss and Accuracy on the validation set. Confusion Matrix as a color mesh on the bottom.

Conclusion: As already mentioned model trained with a balanced dataset has better generalisation (less overfitting on training) and achieved better accuracy on the validation set. It is hard to compare and draw significant conclusions from the confusion matrices but we can observe the values at the diagonals to be slightly highlighted(lighter colors) suggesting the models are working in the right direction. (We can observe a good improvement for the best model in the bonus exercise).

A Better Model

A random search for some of the hyper parameters was performed. The model trained with the parameters mentioned in Table 3 performed the best (with no additional improvements from the bonus exercise).

Parameters	**Value**
N_1	50
K_1	7
N_2	50
K_2	3
ETA	0.01
EPOCH	20,000
ρ	0.9
Seed	112

Table 4: Parameters used for training the model.

The model achieved an accuracy of 51.98% on the validation set which is an improvement of $\sim 5.56\%$ from the earlier models.

	Training Set	**Validation Set**
Accuracy	0.7225	0.5198

Table 5: Final Accuracy on validation set after 20,000 steps.

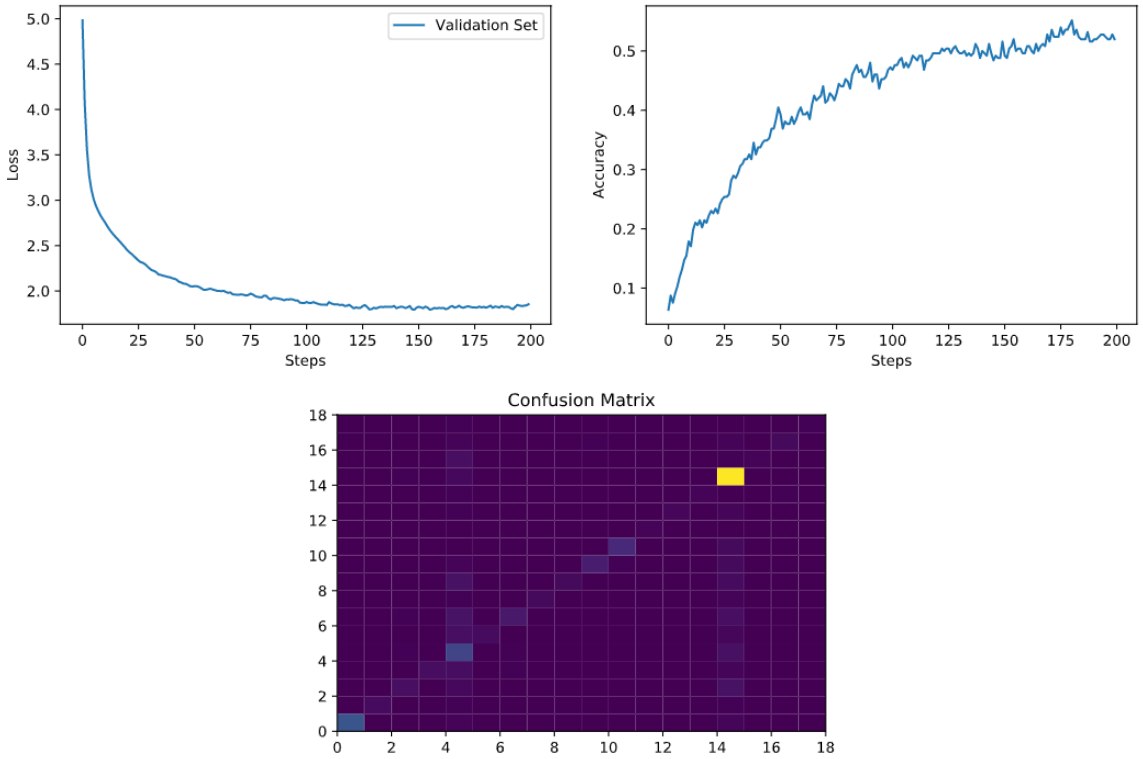


Figure 3: Confusion Matrix as a color mesh on the bottom.

Testing the model

The above model was tested with the last names of some celebrities taken from the web. The results are summarized in the Table 6

Last Name	Actual Label	Predicted Label	Probability
malek	Arabic	Czech	0.435352494
hayek	Arabic	Polish	0.350377179
yang	Chinese	Chinese	0.518403687
yifeng	Chinese	Chinese	0.554321056
kemr	Czech	Scottish	0.200383161
cumberbatch	English	German	0.514555698
sheen	English	Dutch	0.556796881
depardieu	French	French	0.886514104
belmondo	French	Spanish	0.769181824
waltz	German	German	0.384103468
farrell	Greek	English	0.262411198
desica	Italian	Italian	0.531623416
mashkov	Russian	Russian	0.998927451
cruz	Spanish	German	0.177367686
mcgregor	Scottish	Scottish	0.832245552

Table 6: Results on test data.

Conclusion: The model achieved an accuracy of $\sim 46.66\%$ on the above data.From the probabilities in Table 6 ,we can see the prediction for Russian are very accurate. This can be attributed to the significantly high number of training data available from this category. The prediction for Chinese, French , Scottish , Italian are also accurate with good scores. The model seems to struggle with Spanish names a lot .(belmondo $\sim 0.76\%$ and cruz with $\sim 0.17\%$).

Bonus Exercise

Including Bias for all layers

In this exercise, i include the bias term for all the hidden layers. The bias was initialized with $\mathcal{N}$(0,0.001).The gradients were checked with the numerical approximations for a single layered network for one batch and the mean absolute difference between the analytical and numerical gradient was in the range of $\sim 10^{-4} \sim 10^{-5}$. Each of the models presented in the above exercises were trained with the implemented bias. The results are summarized in Table 7.

	Accuracy (Training Set)		**Accuracy(Validation Set)**	
	Without Bias	With Bias	Without Bias	With Bias
Model 1	0.8211	0.7646	0.3690	0.3134
Model 2	0.5925	0.5820	0.4642	0.4523
Model 3	0.7225	0.7001	0.5198	0.5

Table 7: Model Performance after implementing bias in each hidden layer.

Conclusion: We can see that there is a reduction in accuracy on the validation set($2\% \sim 5\%$) on the validation set. However the model is not over fitting the training data as much.In further sections, we tune the parameters of the model that significantly improve the performance of the model.

Generic Code for L layered convolutions

In this exercise, i implement a generic code to apply *l* layered convolutions. The number of layers,number of filters at each layer and the size of the filter can be provided at the initialisation of the CNN class object.The final layer is fully connected .Each layer in the network has a bias term (initialized from a Gaussian distribution) implemented in the previous task. He initialization is used to initialize the weights.The implementation was tested by applying a 2 layered convolutions with the parameters specified in the previous exercises. Similar accuracies were achieved to my prior implementation.I trained several models with different number of convolutional layers.The results from some of the models are summarized in Table 8

Layers	**Number of Filters**	**Filter Size**	**Training Accuracy**	**Validation Accuracy**	**Accuracy on Celebrity Last Names**
1	[50]	[3]	0.7477	0.5238	0.33
2	[20,50]	[5,3]	0.7851	0.4920	0.40
3	[50,50,50]	[7,5,3]	0.8260	0.4642	0.40
5	[50,50,50,50,50]	[5,3,3,3,3]	0.7502	0.4404	0.33
7	[20,20,20,20,20,20]	[3,3,3,3,3,3,3]	0.6323	0.4563	0.26

Table 8: Performance on models with different number of hidden layers.

Conclusion:A general trend can be observed, the accuracy on the validation set seems suffer as we increase the number of hidden layer in the network when working with the given dataset. The model with a single convolution layer outperformed the models specified in the assignment before.

Cyclic ETA

In this exercise , i implemented a cyclic ETA as seen in Figure 4 to work on the multi layered CNN with a specified step size.The parameters ETA_{min} and ETA_{max} were optimized by using the LR range test as implemented in bonus exercise of assignment 2 [**1**].

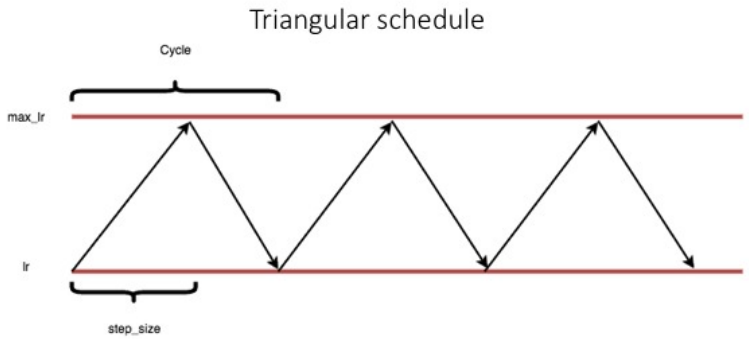


Figure 4: Triangular schedule with no delay

Conclusion: The advantages of using Cyclic ETA have been discussed and analysed in great detail in Assignment 2. The best model trained in the last section has Cyclic ETA implemented along with additional enhancements.

Dropout

For a hidden layer with n nodes, the expectation of the number of neuron to be active at each Dropout is $p * n$, where p is the probability we drop the neuron. These dropped neuron won't propagate anything to the rest of the network.Since we force the network to train with only random p*n of neurons, then intuitively, we force it to learn the data with different kind of neurons subset. This results in better generality of the model.
Conclusion: This was implemented along with the re scaling of the active neurons but it resulted in poor results despite training for more epochs on deeper networks.Upon some investigation [**2**], it was found that dropout is usually applied only on the fully connected layers when working with CNN.This resulted in some improvements with Drop probability of $0.1 \sim 0.2$.

Random Shuffling

Before each epoch the training examples were randomly shuffled.This is done because permuting the training data gives an unbiased estimate of the true gradient, that could improve the accuracy of the model.

Combining all improvements

The results from best performing model with the highest accuracy are summarized below. The model achieved an accuracy of 56.47% which is a relative increase of $\sim 8.63\%$ from earlier models.

Accuracy		
Training Set	**Validation Set**	**Test Set(Celebrity Last Names)**
0.8272	0.5647	0.5333

Parameters	**Value**
Layers	2
Number of Filters	[50,50]
Filter Size	[7,3]
Batch Size	100
Epoch	20,000
Cyclic ETA	0.001-0.1
Step Size	500
Dropout	0
Seed	112
ρ	0.9

Table 9: Parameters used for training the model.

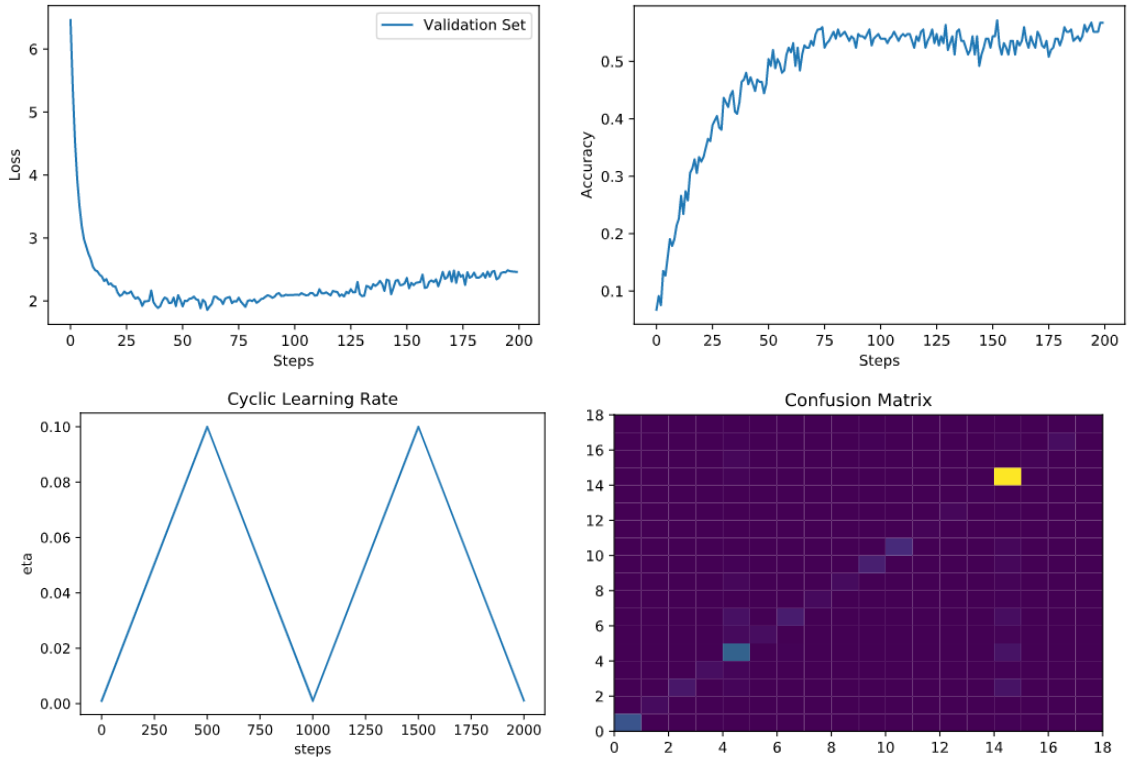


Figure 5: Posterior distribution over W (top) and samples of functions drawn (bottom)

The model was also tested on the test names. Accuracy of 53.33% was achieved which is a relative gain of $\sim 14.29\%$.

Last Name	Actual Label	Predicted Label	Probability
malek	Arabic	Czech	0.837656381
hayek	Arabic	Czech	0.651799388
yang	Chinese	Korean	0.760118819
yifeng	Chinese	Chinese	0.993082727
kemr	Czech	Czech	0.408098922
cumberbatch	English	Irish	0.834496504
sheen	English	Chinese	0.924306473
depardieu	French	French	0.967648267
belmondo	French	Italian	0.306665861
waltz	German	German	0.94551008
farrell	Greek	Irish	0.776229083
desica	Italian	Italian	0.821553079
mashkov	Russian	Russian	0.999700084
cruz	Spanish	Spanish	0.927362053
mcgregor	Scottish	Scottish	0.985334107

References

[1] Smith, L. N. (2015). Cyclical learning rates for training neural networks. arXiv:1506.01186 [cs.CV].

[2] https://www.reddit.com/r/MachineLearning/comments/42nnpe/why_do_i_never_see_dropout_applied_in/