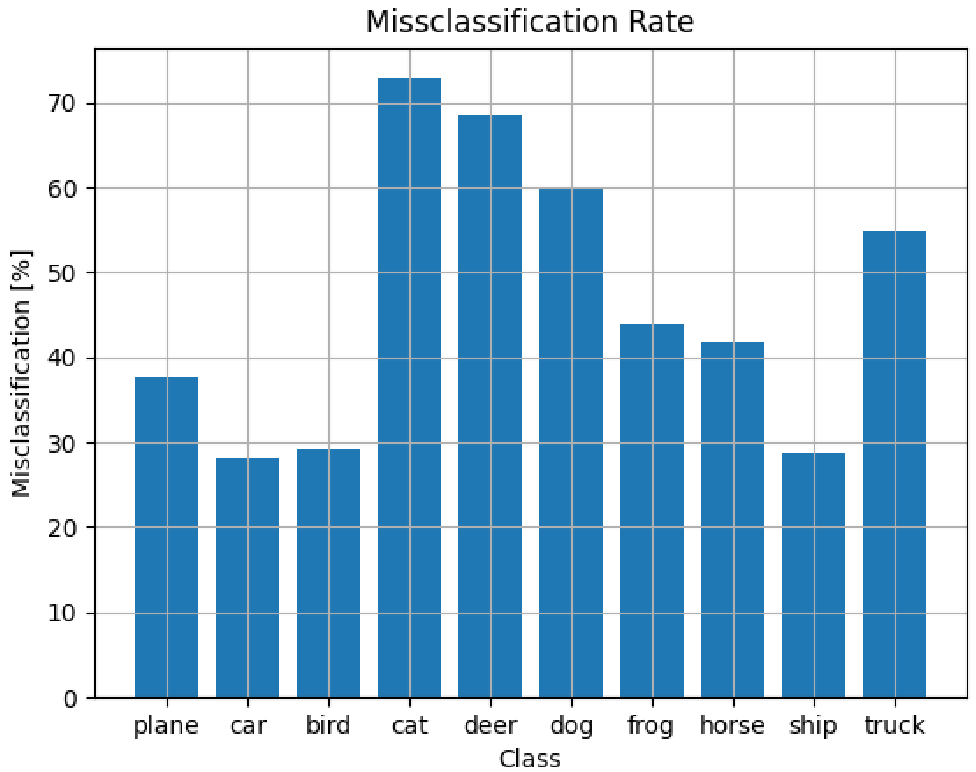
Sigurður Ágúst Jakobsson

Project: Assignment 6 - Applying Neural Networks

Section 1.1

1. How well does your classification work? Plot the misclassification rate for each category onto the same plot.

   The overall accuracy is 53%. This is decent for 10 classes since a guess would lead to 10% accuracy on average like the PyTorch documentation points out. The classification is not as good for all classes. The most misclassification is in the 4 legged mammal classes.



2. Calculate the confusion matrix for the image classification task.

   Confusion matrix (rows = actual, columns = prediction)

[623	19	170	11	13	8	8	11	118	19]
[77	719	24	5	7	5	16	12	59	76]
[65	6	707	33	51	55	32	29	19	3]
[35	16	291	272	85	133	76	49	29	14]
[41	12	431	36	316	24	56	69	14	1]
[17	6	290	132	37	401	29	69	14	5]
[9	13	203	56	101	26	562	13	14	3]
[23	10	193	29	68	72	9	582	8	6]
[142	49	48	9	3	8	3	5	712	21]
[72	209	43	19	16	12	25	80	72	452]

3. Explain what Autograd is and how it works?

   Autograd is a differentiation engine according to the PyTorch documentation. As I understand it, it is a data structure that stores the gradients for all model parameters and allows them to be tracked back in time

Section 2.1

1. What is RNN?

   An RNN is a recurrent neural network. It has some sort of delay function that passes output in one layer back to another, often the input layer. This means that the input of subsequent calculations in the network is affected by earlier calculations/values, and it can retain some history.

2. Why do we use RNN when we are working with text?

   When working with text, we are working with a series of letters that depend on each other in some context. Here previous letters influence the likelihood of what letter will come next. Therefore, since one letter at a time is predicted they are passed back to affect the next guesses to produce something that is not a random jumble of letters. There is a history element in text.

3. In your opinion, how well does the text generation work?

I think it works decently. The output given in the tutorial is the following for a few languages:

Russian: Roveris, Uantovev, Sharinov

German: Gerter, Erenger, Romer

Spaninsh: Sarera, Perere, Arane

Chinese: Cang, Hang, Iun

My output running their code was:

Russian: Rover, Uarin, Shavane

German: Gane, Eren, Roure

Spaninsh: Sanara, Palla, Arana

Chinese: Cha, Han, Iun

On a very rudimentary level the predictions look like something that one might expect in some names in these languages. Römer is actually a german last name. They are probably not completely accurate but definitely manage to emulate some features from the languages.

4. Name three other domains where RNNs are suitable model types for regression/classification

   The first thing that came to my mind was any time series analysis. This could extend to stock price analysis and regression for example, since this is a time series. Then I had to look it up and will reference Wikipedia for this: https://en.wikipedia.org/wiki/Recurrent_neural_network#Applications

   Time series prediction is mentioned there so that is the first domain (and a big one). They mention grammar learning and speech recognition which are closely tied to this text prediction in my mind.

   Somewhat different domains mentioned there are **robot control** and **music composition**. These make sense since robot control is dependent on some history of actions and states. Music composition also makes sense in a way because music has a large time element in what sequence and patterns of notes work well together to create a listenable piece.

Independent Section

Since running the training takes a while and a lot of the time budget for these projects went to project 5 I decided to do some simple tests here. I also don't have CUDA – I understand this is a downloadable toolkit from NVIDIA that allows GPU's to be used to speed up calculations if they are available.

I wanted to test if more epochs of image classification training would improve the accuracy of the image classification since only two epochs were used in the tutorial. I was also interested to see how the surname generation would work on Icelandic. The code is therefore mostly PyTorch tutorial code and not my own, just adjusted for these tasks.

Increasing the epochs from 2 to 20 in the image classification provided the following results:

[641 29 52 36 23 8 11 15 127 58]

[27 771 2 14 5 7 10 10 50 104]

[91 12 464 71 116 95 56 53 21 21]

[41 19 64 388 79 219 65 57 21 47]

[39 6 62 75 577 60 50 102 15 14]

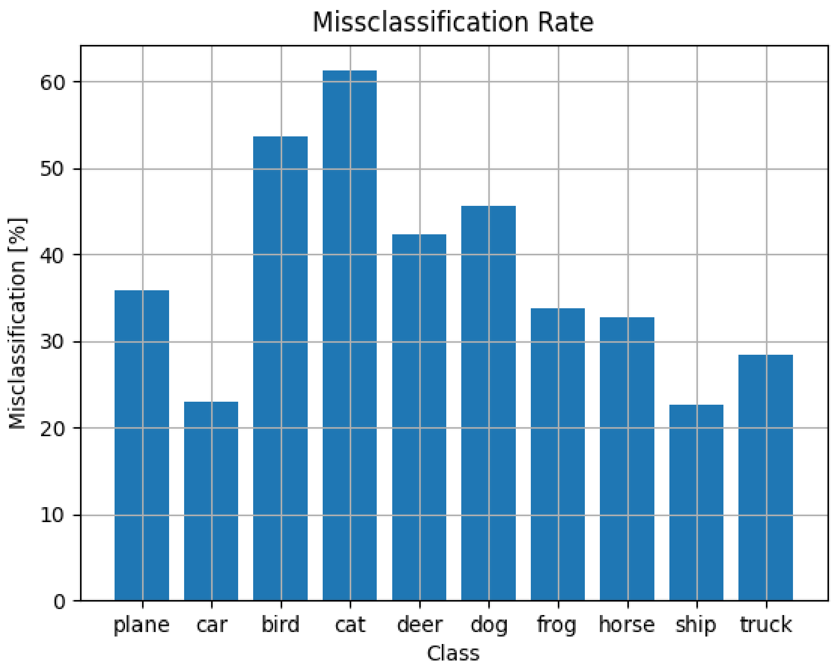
[20 7 66 139 73 544 26 77 11 37]

[14 9 46 79 86 49 663 23 11 20]

[32 5 27 54 76 81 5 673 5 42]

[77 33 14 14 17 5 5 4 774 57]

[55 97 7 20 14 14 5 31 40 717]



Overall accuracy went from 53% to 62% and misclassification dropped quite a bit in certain groups. The highest misclassification was still in the same type of classes though. A tenfold increase in training only improved overall accuracy by about 9%. This is interesting and there is a marginal drop off in the efficiency of additional training it seems. This took about 20 minutes to run so it would be interesting to see how much training would be needed for 90% accuracy for instance.

For the names I went and found a list from https://kidadl.com/baby-names/inspiration/icelandic-last-names-with-meanings-and-history for a list of just under 100 surnames. This was a quick way to find decent data. I cleaned the data up to isolate the names and fix some that had grammar mistakes or didn't make sense. It was interesting that it also had family names, not just patronyms or matronyms. I increased the number of iterations from 100.000 to 300.000 since early tests resulted in nothing that was purely Icelandic but had some patterns that looked Icelandic.

Surname generation from A-Z with 100.000 training iterations generated the following names:

Alassont, Bangandottir, Chandons, Dongansson, Eranssson, Fangandottir, Ganandottir, Hangandot, Irassont, Janansson, Kanasson, Landandoto, Maransson, Nanasson, Oransson, Perandott, Quransson, Ronganssson, Sangans, Trandontorttir, Uananssont, Vangandottir, Warssont, Xangans, Yangandottir, Zangandottir

Surname generation from A-Z with 300.000 training iterations generated the following names:

Argansdo, Brinnsdottiri, Crisson, Dellasson, Erisson, Frinsson, Grinsson, Hangasdottir, Irgansson, Jangasssnn, Kristonsson, Linginss, Marsson, Nigarsson, Olisdontir, Pangasdsti, Quinssson, Rogandsson, Sterasson, Trisson, Uriston, Vingadss, Wallsson, Xrigasson, Yangasss, Ziellsson

This looks a little better to the eye test. It gets the sson and sdottir patterns reasonably well and there is less gibberish in the first part of names. Erisson is one away from Eriksson for instance with a number of other ones close to being spelled correctly. Of course there are letters in there not common to the Icelandic language such as C, X, Y, Q. It would be interesting to see what would happen with more names in the training dataset. It seems that data from other languages might seep in a bit when the model is asked to generate something that isn't typically Icelandic.

The code is turned in with the assignment.

The list in my text training file Icelandic.txt was:

Árnason

Árnadóttir

Ásgeirsdóttir

Birgisdóttir

Birgisson

Bjarnadóttir

Bjarnason

Björnsdóttir

Björnsson

Einarsdóttir

Einarsson

Gísladóttir

Gíslason

Guðjónsdóttir

Guðmundsdóttir

Guðmundsson

Guðjónsson

Gunnarsdóttir

Gunnarsson

Halldórsdóttir

Halldórsson

Harðardóttir

Haraldsson

Hauksson

Helgadóttir

Helgason

Jóhannesdóttir

Jóhannesson

Jóhannsdóttir

Jóhannsson

Jónsdóttir

Jónsson

Karlsdóttir

Karlsson

Kristinsdóttir

Kristinsson

Kristjánsdóttir

Kristjánsson

Magnúsdóttir

Magnússon

Ólafsson

Ólafsdóttir

Óskarsdóttir

Óskarsson

Pálsdóttir

Pálsson

Pétursdóttir

Pétursson

Þórðardóttir

Þorsteinsdóttir

Þorsteinsson

Ragnarsdóttir

Ragnarsson

Sigurjónsdóttir

Stefánsdóttir

Stefánsson

Sveinsdóttir

Sveinsson

Agnarsson

Albertsson

Ármannsson

Heimisson

Hilmarsson

Ingólfsson

Jóhannsson

Leifsson

Þórirsson

Róbertsson

Sigurðsson

Stefánsson

Steinsson

Tómasson

Vilhjálmsson

Alexandersdóttir

Önnudóttir

Aronsdóttir

Baltasarsdóttir

Gudmundardóttir

Guðrúnardóttir

Hauksdóttir

Jónsdóttir

Mikaelsdóttir

Tristandóttir

Beckn

Blöndal

Briem

Egilson

Kemp

Ottesen

Rafnar

Scheving

Stephensen

Thorlacius

Vídalín

Zoëga