Convolutional Neural Network

Machine Learning for Natural Language Processing, Lab Report III
Attention: Required answers are marked in blue !

November 13, 2022

Convolutional neural network is powerful in feature extraction since it is able to identify features regardless of their position. Besides, t extract complex patterns by introducing non-linearity into the model. After the feature extraction, the output can then be sent to a fully-connect network for classification. It is also an efficient algorithm due to the parameter sharing. In this report, binary emotion classification with convolutional neural network will be presented.

1 Emotion Classification

The purpose of emotion classification is to classify tweets from twitter into classes of sentiments including anger, joy, optimism and sadness. This can be achieved by implementing supervised learning algorithm. In this task, convolutional neural network (CNN) will be used for the classification(Baad, n.d.).

2 Convolutional Neural Network (CNN)

Convolutional neural network(Pokharna, n.d.)(Dertat, n.d.)(Prabhu, n.d.)(Afshine Amidi, n.d.) is an efficient algorithm for extracting features. It is composed of several convolutional layers(Dharmaraj, n.d.)(Yadav, n.d.), which act like filters across the input. The features are first extracted and then sent to the fully connected layers(Leonel, n.d.) for classification. Comparing to the multi-layer perceptron (MLP), it is more efficient for its parameter sharing.

2.1 Parameter Sharing

The parameter sharing(Singhal, n.d.)(Alake, n.d.) in CNN makes it significantly more efficient than MLP. This is because it reduces the parameters by applying the convolutional layers.

Take a one layer neural network for instance. Let's say the input dimension is D_i, and the hidden layer has dimension D_h. For MLP, the number of parameters is $D_i \times D_h$. However, for CNN, the parameters are shared by passing kernels through windows. This significant reduction in memory and time resources can be validated by being able to run the task in CPU.

3 Method

The data is first preprocessed to eliminate redundant words. This will make the training faster and more efficient. Then each word will be indexed and sent to the convolutional neural network for training. The performance of three models will be tested by the validation datasets. The best performance model will then be evaluated by the testing dataset.

3.1 Data

The data in the repository is pairs of tweets from Twitter and the corresponding emotions as labels. It is split into training, validation, and testing datasets. There are four emotions available, which are anger, joy, optimism and sadness, and each of them are labeled as 1, 2, 3 and 4 respectively.

3.1.1 Cloning & Saving Dataset

The data for this task can be obtained from Github Repository. In the repository, there are several directories, each for the purpose of a different task. The "emotion" directory is the one that is used in the task.

Since the data is large, downloading it locally and uploading it to Colab might not be the most efficient way. Thus, the repository is cloned directly from the Github repository.

After the repository is cloned to Google Drive, redundant folders can be deleted. Training, validation and

Table 1: *Labels Conversion*

Task	Label 0	Label 1
joy & sadness	sadness	joy
anger & sadness	anger	sadness

Table 2: *Batch Size*

Dataset	Batch Size
Train	16
Validation	8
Test	16

testing data in the emotion folder can then be further processed.

3.1.2 Choosing Emotions

Two pairs of emotions will be chosen for the task. The first pair of emotion, joy and sadness, will be used to train the model. The best performing architecture will be used to trained the second pair of emotion, anger and sadness, for comparison.

3.1.3 Data Preprocessing

The purpose of data preprocessing is to eliminate redundant words to reduce the dimension of input for CNN. The data preproccessing in this task includes

1. lowercasing
2. removing HTML tags and URLs
3. removing emojis
4. removing punctuation and numbers
5. removing single character words

The methods above can significantly reduce the dimension of the input tweets, and at the same time, not affecting the performances since those symbols or words do not have much contributions toward the semantics of the tweet.

3.1.4 Label Preprocessing

The CNN will be done by `PyTorch` for this task. It allows labels starting from 0, and thus, the labels in joy & sadness will be converted from 1 & 3 to 1 & 0. And for the task of anger & sadness, the labels will be converted from 0 & 3 to 0 & 1 as shown in Table 1.

3.2 Tokenising & Encoding

Before putting tweets into CNN, the context should be transformed into numeric tensors. Thus, the tweets will first be tokenised, and then encoded.

The tweets are tokenised into words. Each words are given an unique index starting from 2. The index 0 is reserved for paddings to ensure the input is a tensor that has same length for all sentences. This means that any tweets shorter than the longest tweet will be padded to the same length as the longest one. The index 1 is reserved for unknown words appearing in validation or test datasets but not in training dataset. The word-index pairs will be store in a dictionary, `word2idx`.

By looking up the `word2idx` dictionary, each tweets can be easily encoded with paddings in the end. The `word2idx` generated by the train dataset will be used to encode the validation and testing datasets.

3.3 Batch Preparation

Before putting data into training, data should be first split into batches. This can be done by `PyTorch` DataLoader. In the task, the batch size of each dataset is shown in Table 2.

3.4 CNN Training

In the training process(Rosebrock, n.d.), a total of eight models are trained for joy & sadness dataset. The best performing model architecture will be used to train the anger & sadness dataset. To see the architecture used in all eight models, please refer to Colab notebook or the end of the report for more infomation.

The training dataset will be used for training purpose, and the validation dataset will be used to evaluate each model architecture. The best performing model will then be tested on the test dataset for accuracy performance.

Each dataset will be trained for 50 epochs. Whenever the accuracy exceeds the current best accuracy, the model is saved(Quinn Radich, n.d.). By this way, the best model without overfitting will be saved.

3.4.1 Joy & Sadness Dataset Training

For the joy & sadness dataset training, three models will be listed out for discussion. All models use Adam with decay learning rate as the optimiser and cross entropy loss for the loss function(Wang, n.d.). For each model, classification report(Verma, n.d.) from `PyTorch` will be printed out.

Model 1 is composed of an embedding layer with dimension 50, followed by 25 convolutional layers with dropout probability equals 0.2. The output is then sent to a fully-connect layer with dropout prabability equals 0.5 for classification. Model 1 has the following architecture listed in Table 3.

Table 3: *joy & sadness Model 1 architecture*

Layer	Details
input	
(b_size, max_len)	(16, 33)
embedding	
(b_size, max_len, emb_dim)	(16, 33, 50)
convolution	
filter_size	3×20+5×5
non-linear	ReLu
drop_out	0.2
num_filter	50×25
stride	1×20+3×2+4×3
padding	None
Dilation	None
fully-connect	
linear	
drop_out	0.5
Sigmoid	
optimiser	Adam
learning rate	0.0001
decay rate	0.00001
loss function	cross entropy loss

Table 4: *joy & sadness Model 2 architecture*

Layer	Details
input	
(b_size, max_len)	(16, 33)
embedding	
(b_size, max_len, emb_dim)	(16, 33, 300)
convolution	
filter_size	3×20+4×20+5×10
non-linear	ReLu
drop_out	0.2
num_filter	100×30+50×20
stride	1×15+3×20+4×10
padding	None
Dilation	None
fully-connect	
linear	
drop_out	0.5
Sigmoid	
optimiser	Adam
learning rate	0.0001
decay rate	0.00001
loss function	cross entropy loss

Model 2 is composed of an embedding layer with dimension 300, followed by 50 convolutional layers with dropout probability equals 0.2. The result is also sent to a fully-connected layer with dropout prabability equals 0.5 for classification. Model 2 has architecture listed in Table 4.

Model 8 is composed of an embedding layer with dimension 300, followed by 30 convolutional layers without dropout. The result is also sent to a fully-connected layer with dropout prabability equals 0.5 for classification. Model 8 has architecture listed in Table 5.

Table 5: *joy & sadness Model 8 architecture*

Layer	Details
input	
(b_size, max_len)	(128, 33)
embedding	
(b_size, max_len, emb_dim)	(128, 33, 300)
convolution	
filter_size	3×25+5×5
non-linear	ReLu
drop_out	None
num_filter	50×30
stride	1×20+3×2+4×3
padding	None
Dilation	None
fully-connect	
linear	
drop_out	0.5
Sigmoid	
optimiser	Adam
learning rate	0.0001
decay rate	0.00001
loss function	cross entropy loss

3.4.2 Anger & Sadness Dataset Training

The best performing model architecture from joy & sadness dataset training will be used to train the anger & sadness dataset. The best performing model is model 2 with embedding dimension of 300 and dropout layers of probability 0.2 between the convolution layers.

4 Result

After training, loss and accuracy for each epoch(Viswarupan, n.d.) are printed out for both training and validation datasets.

Table 6: *Joy & Sadness Model 1 Result*

Model 1 Result Summary	
Train batch size	16
Accuracy	69.35%
Epoch saved at	49
Total parameters	475'952
Loss for each epoch	Figure 1
Accuracy for each epoch	Figure 2
Confusion matrix	Figure 3

Table 7: *Joy & Sadness Model 2 Result*

Model 2 Result Summary	
Train batch size	16
Accuracy	75.26%
Epoch saved at	46
Total parameters	5'920'202
Loss for each epoch	Figure 4
Accuracy for each epoch	Figure 5
Confusion matrix	Figure 6

4.1 Joy & Sadness Dataset Result

For Model 1, Model 2, and Model 3, the results are shown in Table 6, Table 7, Table 8 respectively. In the loss and accuracy epoch plot, the green vertical line indicates the point where the model is saved. The testing result is shown in Table 9.

4.2 Anger & Sadness Dataset Result

The best model, model 2, is used for training this dataset. Only the testing result will be shown for this dataset, and the testing result is shown in Table 10. For the result of training process, please refer to Colab notebook.

Table 8: *Joy & Sadness Model 8 Result*

Model 8 Result Summary	
Train batch size	128
Accuracy	66.66%
Epoch saved at	39
Total parameters	3'062'702
Loss for each epoch	Figure 7
Accuracy for each epoch	Figure 8
Confusion matrix	Figure 9

Table 9: *Joy & Sadness Model 2 Testing Result*

Test Summary	
Train batch size	16
Accuracy	76.89%
Confusion matrix	Figure 10
F1-macro	0.7661

Table 10: *Anger & Sadness Model 2 Testing Result*

Test Summary	
Train batch size	16
Accuracy	78.94%
Confusion matrix	Figure 11
F1-macro	0.7113

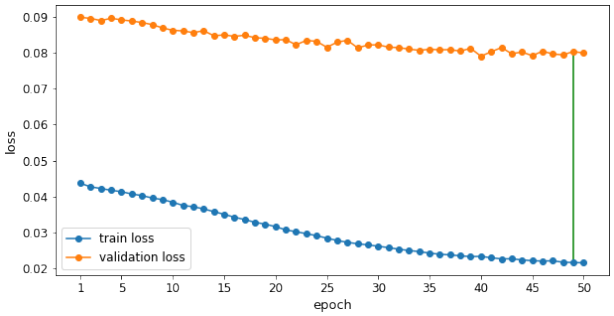

Figure 1: *Model 1 Loss*

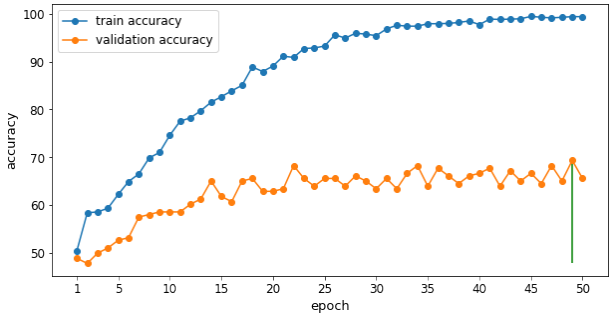

Figure 2: *Model 1 Accuracy*

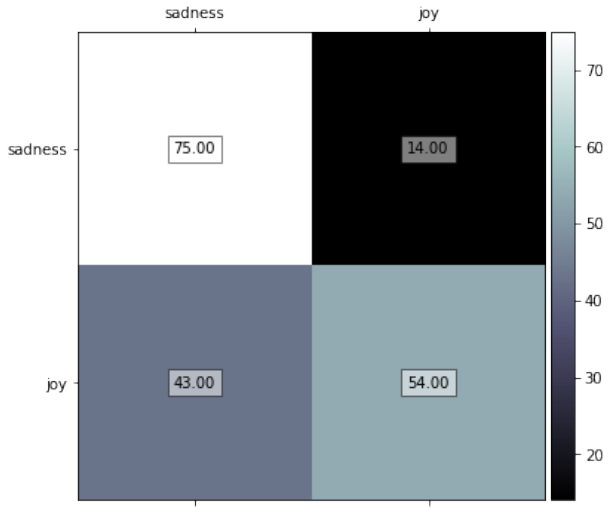

Figure 3: *Model 1 Confusion Matrix*

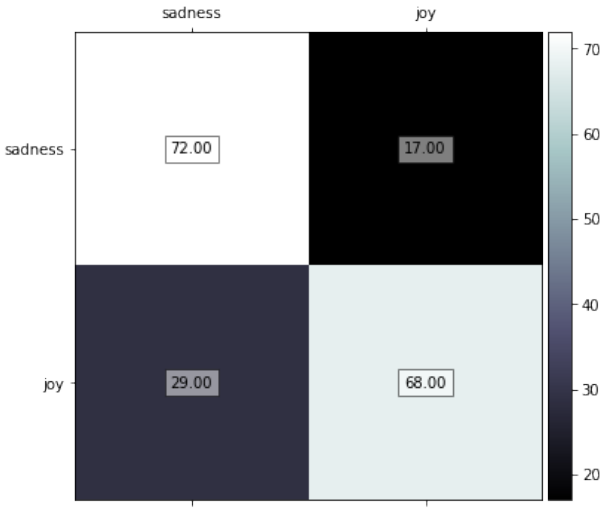

Figure 6: *Model 2 Confusion Matrix*

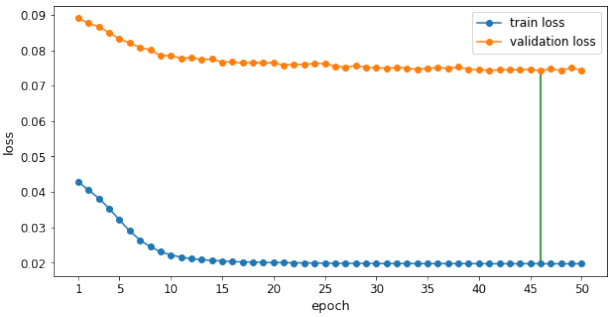

Figure 4: *Model 2 Loss*

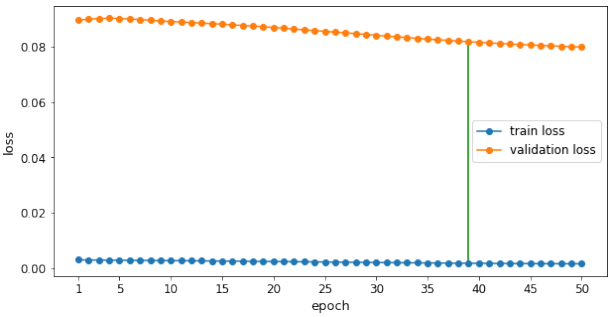

Figure 7: *Model 8 Loss*

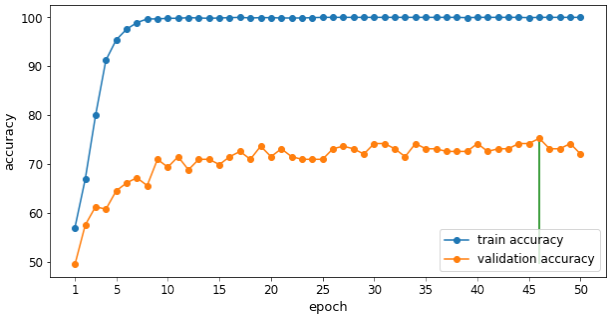

Figure 5: *Model 2 Accuracy*

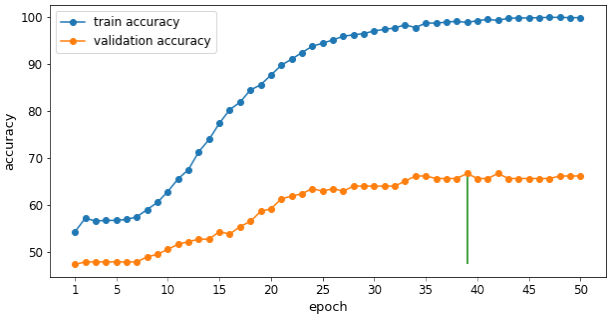

Figure 8: *Model 8 Accuracy*

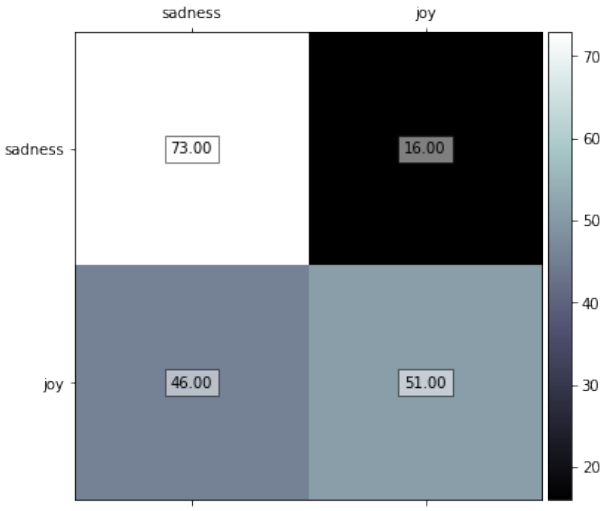

Figure 9: *Model 8 Confusion Matrix*

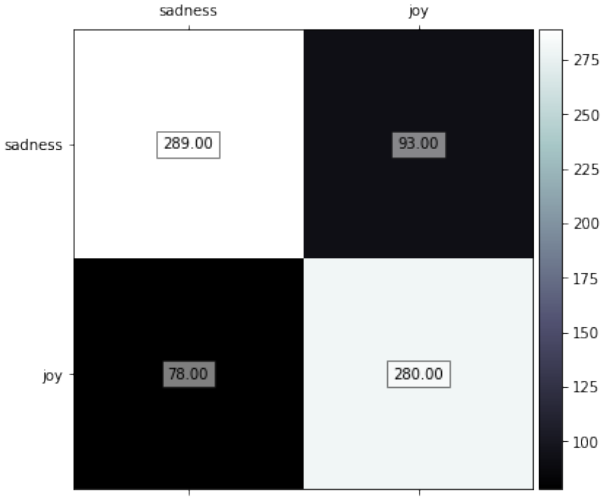

Figure 10: *Joy & Sadness Testing Confusion Matrix*

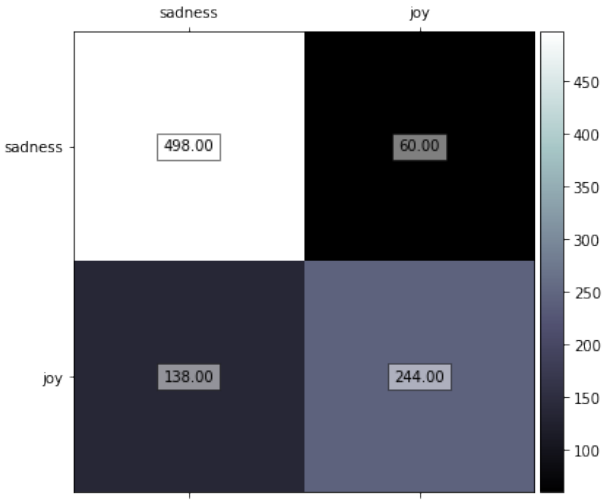

Figure 11: *Anger & Sadness Testing Confusion Matrix*

5 Discussion

The table in the appendix listed out all models tried. It summarizes the 8 different sets of parameters we tested when training the models on the joy & sadness dataset and their corresponding accuracy scores. One of the first parameters that we changed was embedding dimension. In Model 1, the embedding dimension was 50, whereas in the rest of the models it was 300. Choosing the right embedding dimension is crucial for NLP models, as too low embedding dimension does not capture enough information, resulting in very limited model performance, whereas too large embedding dimension can be computationally very expensive and can also lead to model overfitting, resulting in poor accuracy score on the test data. The increase of embedding dimension (from 50 to 300) can explain why the performance of Model 1 is worse than performance of Models 2 to 7.

Next, we also changed the number of filters (or kernels) and filter size. Filter size indicates the number of input elements (tokens) that a convolution takes at each step. The number of filters parameter corresponds to the number of output features, as each filter in a convolution extracts 1 feature. For our dataset, we can see from the table that changing the filter size and numbers did not have a large effect on performance accuracy. For example, Models 3 and 6 differ only in filter size, and they have comparable accuracy scores.

The stride parameter indicates the magnitude of the filter movement at each step, i.e. a stride of 1 means that the filter moves by 1 token at each step. Looking at our results, it seems that it does not have a large effect on the model performance.

The padding parameter is adding zero vectors to both sides of the input token. In the context of NLP, we may need to include padding since every sentence in the text does not have the same number of words, which has to be accounted for, as all the neural networks require to have inputs that have the same shape and size.

Dropout is a parameter that provides a way to reduce the overfitting of the model. It refers to dropping out the nodes in a neural network. The nodes are dropped with a dropout probability specified by the parameter value. For example, there are 100 neurons in the hidden layer, assigning a dropout of 0.5 means that 50 neurons will be randomly silenced in every iteration. In Models 2 and 4, we included a dropout after each convolutional layer, which seems to improve the model performance, since Models 2 and 4 both have higher prediction accuracy than the rest of the models, which included a dropout only after one of the convolutional layers.

Although not saved in Colab notebook, models with di-

lation is also trained. However, the validation accuracy is very low. This might because the semantic of a text is sequential. For instance, given a sentence, *The cake is not good*. With dilation, the CNN might capture *is good* instead of *is not* and *not good*. This might cause the classification accuracy to suffer.

In Model 8 , we increased the batch size from 16 to 128, which means that the model state gets updated after a higher number of training samples was processed. While it decreases the time it takes to train the model, it also significantly decreases the model performance accuracy, as our results show.

Bibliography

Afshine Amidi, Shervine Amidi (n.d.). *Convolutional Neural Networks cheatsheet*. URL: https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-convolutional-neural-networks. (accessed: 11.11.2022).

Alake, Richmond (n.d.). *Understanding Parameter Sharing (or weights replication) Within Convolutional Neural Networks*. URL: https://towardsdatascience.com/understanding-parameter-sharing-or-weights-replication-within-convolutional-neural-networks-cc26db7b645a. (accessed: 11.11.2022).

Baad, Dipika (n.d.). *Sentiment Classification for Restaurant Reviews using CNN in PyTorch*. URL: https://towardsdatascience.com/sentiment-classification-using-cnn-in-pytorch-fba3c6840430. (accessed: 11.11.2022).

Dertat, Arden (n.d.). *Applied Deep Learning - Part 4: Convolutional Neural Networks*. URL: https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2. (accessed: 11.11.2022).

Dharmaraj (n.d.). *Convolutional Neural Networks (CNN) — Architecture Explained*. URL: https://medium.com/@draj0718/convolutional-neural-networks-cnn-architectures-explained-716fb197b243. (accessed: 11.11.2022).

Leonel, Jorge (n.d.). *Multilayer Perceptron*. URL: https://medium.com/@jorgesleonel/multilayer-perceptron-6c5db6a8dfa3. (accessed: 11.11.2022).

Pokharna, Harsh (n.d.). *The best explanation of Convolutional Neural Networks on the Internet!* URL: https://medium.com/technologymadeeasy/the-best-explanation-of-convolutional-neural-networks-on-the-internet-fbb8b1ad5df8. (accessed: 11.11.2022).

Prabhu (n.d.). *Understanding of Convolutional Neural Network (CNN) — Deep Learning*. URL: https://medium.com/@RaghavPrabhu/understanding-of-convolutional-neural-network-cnn-deep-learning-99760835f148. (accessed: 11.11.2022).

Quinn Radich Mitra Lotfi Shemirani, Alma Jenks (n.d.). *Train your data analysis model with PyTorch*. URL: https://learn.microsoft.com/en-us/windows/ai/windows-ml/tutorials/pytorch-analysis-train-model. (accessed: 11.11.2022).

Rosebrock, Adrian (n.d.). *PyTorch: Training Your First Convolutional Neural Network (CNN)*. URL: https://pyimagesearch.com/2021/07/19/pytorch-training-your-first-convolutional-neural-network-cnn/. (accessed: 11.11.2022).

Singhal, Vishrut (n.d.). *Explain the Significance of Parameter Sharing and Sparsity of Connections in CNN*. URL: https://discuss.boardinfinity.com/t/explain-the-significance-of-parameter-sharing-and-sparsity-of-connections-in-cnn/8552. (accessed: 11.11.2022).

Verma, Akshaj (n.d.). *PyTorch [Tabular] — Binary Classification*. URL: https://towardsdatascience.com/pytorch-tabular-binary-classification-a0368da5bb89. (accessed: 11.11.2022).

Viswarupan, Niruhan (n.d.). *Drawing Loss Curves for Deep Neural Network Training in PyTorch*. URL: https://towardsdatascience.com/pytorch-tabular-binary-classification-a0368da5bb89https://niruhan.medium.com/drawing-loss-curves-for-deep-neural-network-training-in-pytorch-ac617b24c388. (accessed: 11.11.2022).

Wang, Benjamin (n.d.). *Loss Functions in Machine Learning*. URL: https://medium.com/swlh/cross-entropy-loss-in-pytorch-c010faf97bab. (accessed: 11.11.2022).

Yadav, Suraj (n.d.). *In-Depth Knowledge of Convolutional Neural Networks*. URL: https://medium.com/@Suraj_Yadav/in-depth-knowledge-of-convolutional-neural-networks-b4bfff8145ab. (accessed: 11.11.2022).

Model Number	Parameters	Performance
1	Number of classes = 2 Embeddings dimension = 50 Filter size = [3x20] + [5x5] Stride = [1x20] + [3x2] + [4x3] Dropout 1 = 0.2 Padding = [0x25] Dilation = [1x25] Number of filters = [50x25] Dropout 2 = 0.5	Validation accuracy = 69.35% Macro F1-score = 0.65
2	Number of classes = 2 Embeddings dimension = 300 Filter size = [3x20] + [5x5] Stride = [1x15] + [3x10] + [4x5] Dropout 1 = 0.2 Padding = [0x30] Dilation = [1x30] Number of filters = [50x30] Dropout 2 = 0.5	**Validation accuracy = 76.34%** **Macro F1-score = 0.73**
3	Number of classes = 2 Embeddings dimension = 300 Filter size = [3x20] + [5x5] Stride = [1x15] + [3x10] + [4x5] Dropout 1 = 0.5 Padding = [0x30] Dilation = [1x30] Number of filters = [50x30]	Validation accuracy = 73.12% Macro F1-score = 0.70
4	Number of classes = 2 Embeddings dimension = 300 Filter size = [3x20] + [4x20] + [5x10] Stride = [1x15] + [3x20] + [4x10] Dropout 1 = 0.5 Padding = [0x50] Number of filters = [100x30] + [50x20] Dropout 2 = 0.5	Validation accuracy = 75.27% Macro F1-score = 0.72
5	Number of classes = 2 Embeddings dimension = 300 Filter size = [2x15] + [3x5] + [4x5] + [5x5] Stride = [1x15] + [2x10] + [4x5] Dropout 1 = 0.5 Padding = [2x30] Dilation = [1x30] Number of filters = [100x10] + [50x20]	Validation accuracy = 71.51% Macro F1-score = 0.71
6	Number of classes = 2 Embeddings dimension = 300 Filter size = [3x25] + [5x5] Stride = [1x15] + [3x10] + [4x5] Dropout 1 = 0.5 Padding = [0x30] Dilation = [1x30] Number of filters = [50x30]	Validation accuracy = 72.58% Macro F1-score = 0.65
7	Number of classes = 2 Embeddings dimension = 300 Filter size = [3x25] + [5x5] Stride = [1x15] + [3x10] + [4x5] Dropout 1 = 0.5 Padding = [0x30] Dilation = [1x30] Number of filters = [50x30] max pool --> avg pool	Validation accuracy = 73.66% Macro F1-score = 0.72
8	Number of classes = 2	Validation accuracy =

	Embeddings dimension = 300 Filter size = [3x25] + [5x5] Stride = [1x15] + [3x10] + [4x5] Dropout 1 = 0.5 Padding = [0x30] Dilation = [1x30] Number of filters = [50x30] batch size: 16 --> 64	66.67% Macro F1-score = 0.66