Analyzing a CNN-based NLP Model

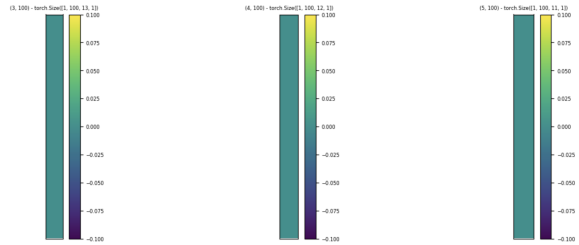
For the NLP experiments we used a CNN architecture described in [1]. The original model does not contain any paddings in the convolutional layers. More details on how to train the model can be found here: https://github.com/bentrevett/pytorch-sentiment-analysis/blob/master/4%20-%20Convolutional%20Sentiment%20Analysis.ipynb

We decided to pad convolutional layers with zero padding along the token dimension axis and compare the performance of the model with and without padding. Table 1 summarizes model performance results for different padding zero padding lengths. For this specific model we didn't observe any significant improvements with and without padding.

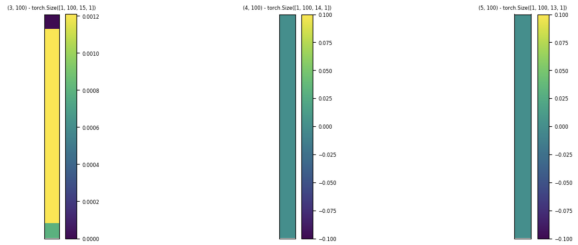
	A	B	C	D	E	F	G	H	I
1	IMDB Training / Testing performance numbers (The network learns to correct the residuals emerged by the paddings), 5 epochs each								
2		No padding	Padding 1	Padding 2	Padding 3	Padding 6			
3	Train Acc	94.33%	94.19%	94.22%	93.92%	93.75%			
4	Valid Acc	86.45%	86.41%	86.31%	86.74%	86.56%			
5	Test Acc	85.69%	85.50%	85.43%	85.77%	85.76%			
6									
7	Table 1: Model performance evaluation results without padding (second column) and with paddings of different lenght (column 3 to 5).								
8									

Similar to computer vision models we visualized the feature maps for uniform input and zero padding and confirmed that the artifacts get propagated inwards.

IMDB - No padding - uniform input

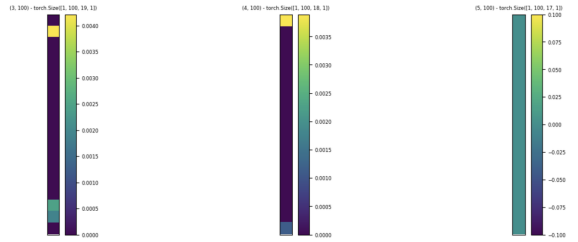


IMDB - Padding1 - the effects of padding - uniform input



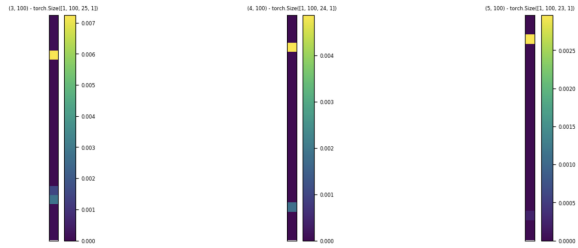
Zero padding artifacts in convolutional layers. In this case we used zero padding of length 1

IMDB - Padding3 - the effects of padding - uniform input



Zero padding artifacts in convolutional layers. In this case we used zero padding of length 3

IMDB - Padding6 - the effects of padding - uniform input



Zero padding artifacts in convolutional layers. In this case we used zero padding of length 6

In addition to that we also pre and post padded the input string and used an attribution algorithm called integrated gradients to find out if the padding has any effect on the attribution quality.

The effects of input pre-padding on the attribution (this one was with conv zero padding 6)

```
Visualize attributions based on Integrated Gradients
```

True Label	Predicted Label	Attribution Label	Attribution Score	Word Importance
pos	pos (0.90)	pos	1.03	#pad #pad #pad #pad #pad #pad This film is great

The effects of input post-padding on the attribution (this one was with conv zero padding 6)

```
Visualize attributions based on Integrated Gradients
```

True Label	Predicted Label	Attribution Label	Attribution Score	Word Importance
pos	pos (1.00)	pos	1.04	This film is great #pad #pad #pad #pad #pad #pad

We show empirically that the input padding type has an effect on the attribution results. As shown in the example above if we pre-pad the input then the first token in the sentence negatively contributes to the prediction but if we post-pad it then it has slight positive attribution to the prediction. It's also interesting to observe that padding has no significant effect on the attribution of the strong predictive token `great` which happen to be the last token in the sentence.

Attribution when there is no padding in conv layers

```
Visualize attributions based on Integrated Gradients
```

True Label	Predicted Label	Attribution Label	Attribution Score	Word Importance
pos	pos (0.99)	pos	0.99	The film is great #pad #pad #pad #pad #pad #pad #pad #pad #pad #pad #pad

---

[1] Yoon (2014): Convolutional Neural Networks for Sentence Classification (https://arxiv.org/pdf/1408.5882.pdf)