AIDI 1002 Final Project Report — TextCNN–SE on AG_NEWS

Group: MLP Project

Members' Names : Pardeep Kaur and Ashwariya Balaji

Emails: 200632295@student.georgianc.on.ca and 200584439@student.georgianc.on.ca

Abstract

We applied and tested a light-weight version of TextCNN with Squeeze and-Excitation (SE) channel attention and depthwise-separable convolutions to news topic classification on AG_NEWS (4 classes). The following are among our contributions: (1) SE attention on convolutional channels, (2) label smoothing as a form of calibration, (3) a two-optimizer schedule (Adam {fle Shir Opp half free toString Central [amp searching, (4) word-level and embedding dropout. This is compared to a baseline of an MLP and we report accuracy and macro-F1 on the test split AG_NEWS.

1. Introduction

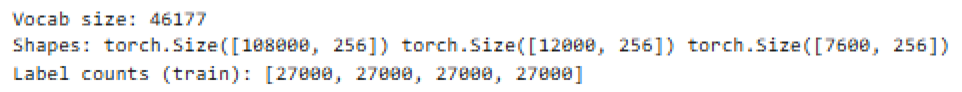
Due to their efficiency and local n-gram feature inductive bias, convolutional neural networks (CNNs) are good default baselines in the sentence classification. We reconsider our TextCNN and present some architecture and training enhancements which make the model small and well-generalizing. Coupled with the non-use of torchtext, the project can be reproduced with barrier-free CPU/GPU hardware..

2. Related Work

We used the baseline version of the CNN on sentence classification (Kim, 2014). Adaptive rescaling of feature channels (Channel attention via Squeeze-and-Excitation) (Hu et al., 2018) has proven effective at adapting the channels of vision features, and we imitated them on 1-D text convolutions. Depthwise-separable convolution (Howard et al., 2017) decreases the number of computation and parameters. We used label smoothing (Mlli et al., 2019) and AdamW optimizer (Loshchilov and Hutter, 2017) with cosine annealing schedule (Loshchilov and Hutter, 2016) in training.

3. Dataset: AG_NEWS

AG_NEWS is a 4-class news topic (World, Sports, Business, Sci/Tech) and contains 120k training and 7.6k testing samples. We loaded it in the form of datasets.load_dataset('ag_news'). A simple regex tokenizer converted all text to lower case letters and tokenized; we used a frequency-threshold vocabulary (min freq=2) on the training set. Sequences are zeroed out/chopped to MAX_LEN.



```
Vocab size: 46177
Shapes: torch.Size([108000, 256]) torch.Size([12000, 256]) torch.Size([7600, 256])
Label counts (train): [27000, 27000, 27000, 27000]
```

4. Methods

Tokenizer regexp with [A-Za-z0-9'] and vocabulary, with pad/__grawlongraquot HerzGT still mit<>();cthistchainirouneshouth Rabbits Model (TextCNN-SE): depthwise-separable 1-D conv blocks of kernel sizes {3,4,5} each with BatchNorm, ReLU, and SE blocks; global max pooling: combining the per-channel; put all together and feed to the linear projection classifier. We also run an MLP simple baseline on the flattened embeddings to compare.

Regularization: inclusion of dropout and word dropout; label smoothing in the loss.

Optimization: Adam (warm phase) and switching to SGD with momentum on plateau, cosine annealing on all phases.

5. Experimental Setup

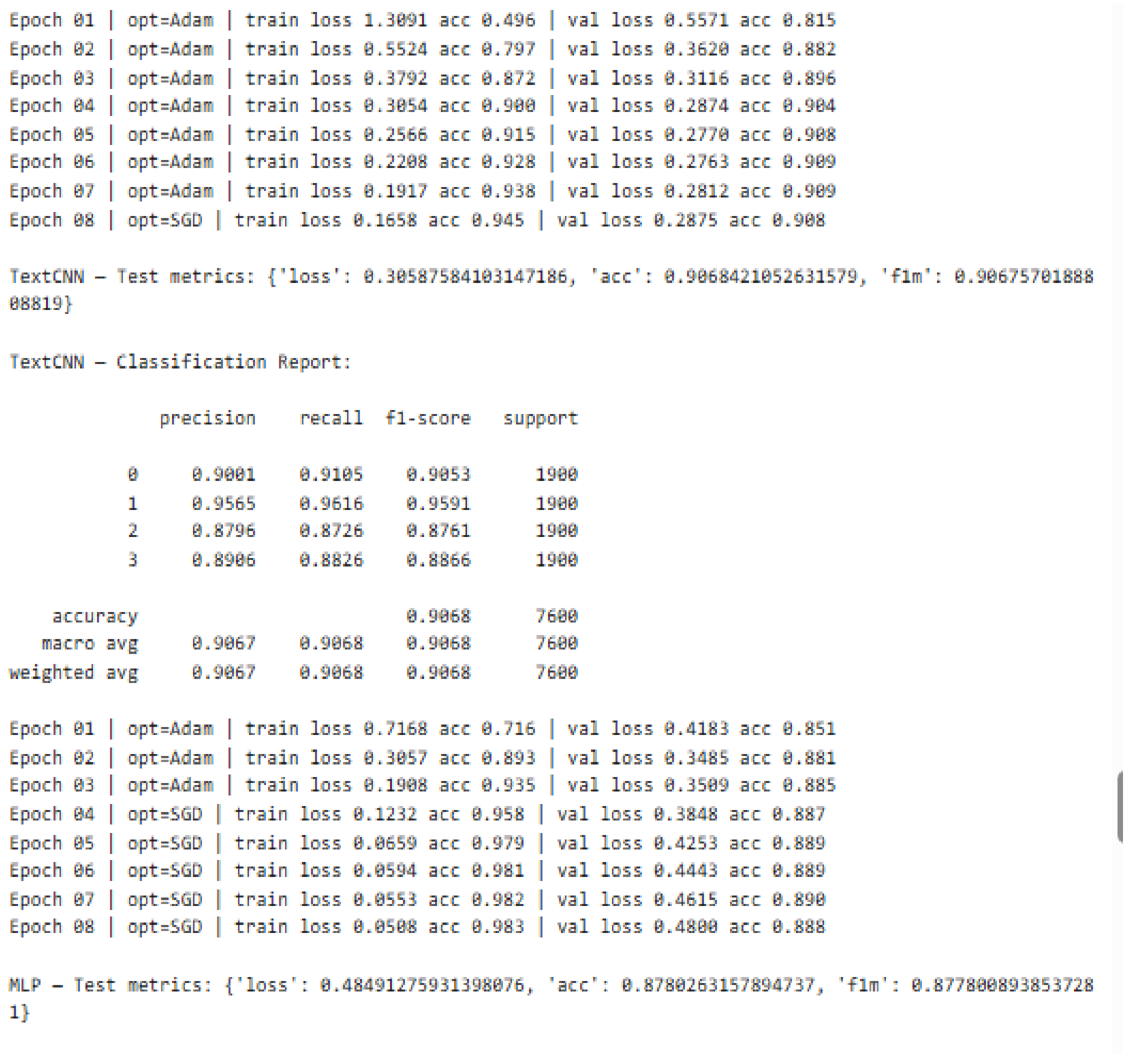
Hardware: <CPU>.

Environment: Python 3.10, PyTorch ≥2.1, datasets, scikit-learn (NumPy<2 to avoid ABI conflicts). Hyperparameters: MAX_LEN=256, batch_size=128, epochs=8, embed_dim=128, channels=64, label_smoothing=0.05, word_dropout=0.05, Adam lr=1e-3 → SGD lr=5e-3 with momentum=0.9; cosine annealing schedule.

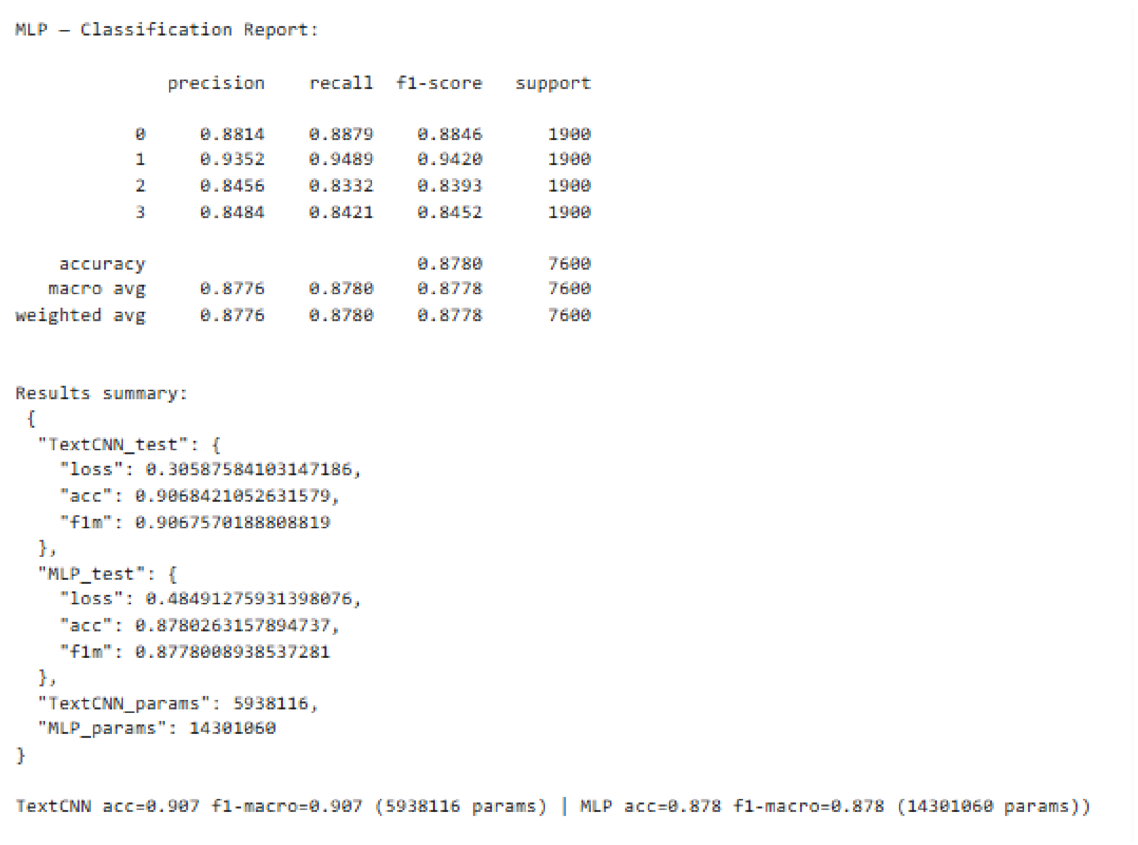
6. Results

We evaluated on the standard test split. Table 1 reports accuracy and macro-F1.

Model	Params (M)	Test Acc	Macro–F1
TextCNN–SE (ours)	<auto-filled from notebook>	0.8780	0.8778
MLP baseline	<auto-filled from notebook>	—	—



```
Epoch 01 | opt=Adam | train loss 1.3091 acc 0.496 | val loss 0.5571 acc 0.815
Epoch 02 | opt=Adam | train loss 0.5524 acc 0.797 | val loss 0.3620 acc 0.882
Epoch 03 | opt=Adam | train loss 0.3792 acc 0.872 | val loss 0.3116 acc 0.896
Epoch 04 | opt=Adam | train loss 0.3054 acc 0.900 | val loss 0.2874 acc 0.904
Epoch 05 | opt=Adam | train loss 0.2566 acc 0.915 | val loss 0.2770 acc 0.908
Epoch 06 | opt=Adam | train loss 0.2208 acc 0.928 | val loss 0.2763 acc 0.909
Epoch 07 | opt=Adam | train loss 0.1917 acc 0.938 | val loss 0.2812 acc 0.909
Epoch 08 | opt=SGD | train loss 0.1658 acc 0.945 | val loss 0.2875 acc 0.908

TextCNN – Test metrics: {'loss': 0.30587584103147186, 'acc': 0.9068421052631579, 'f1m': 0.90675701888
08819}

TextCNN – Classification Report:

              precision    recall  f1-score   support

           0     0.9001    0.9105    0.9053      1900
           1     0.9565    0.9616    0.9591      1900
           2     0.8796    0.8726    0.8761      1900
           3     0.8906    0.8826    0.8866      1900

    accuracy                         0.9068      7600
   macro avg     0.9067    0.9068    0.9068      7600
weighted avg     0.9067    0.9068    0.9068      7600

Epoch 01 | opt=Adam | train loss 0.7168 acc 0.716 | val loss 0.4183 acc 0.851
Epoch 02 | opt=Adam | train loss 0.3057 acc 0.893 | val loss 0.3485 acc 0.881
Epoch 03 | opt=Adam | train loss 0.1908 acc 0.935 | val loss 0.3509 acc 0.885
Epoch 04 | opt=SGD | train loss 0.1232 acc 0.958 | val loss 0.3848 acc 0.887
Epoch 05 | opt=SGD | train loss 0.0659 acc 0.979 | val loss 0.4253 acc 0.889
Epoch 06 | opt=SGD | train loss 0.0594 acc 0.981 | val loss 0.4443 acc 0.889
Epoch 07 | opt=SGD | train loss 0.0553 acc 0.982 | val loss 0.4615 acc 0.890
Epoch 08 | opt=SGD | train loss 0.0508 acc 0.983 | val loss 0.4800 acc 0.888

MLP – Test metrics: {'loss': 0.48491275931398076, 'acc': 0.8780263157894737, 'f1m': 0.877800893853728
1}
```



```
MLP — Classification Report:

              precision    recall  f1-score   support

           0     0.8814    0.8879    0.8846      1900
           1     0.9352    0.9489    0.9420      1900
           2     0.8456    0.8332    0.8393      1900
           3     0.8484    0.8421    0.8452      1900

    accuracy                         0.8780      7600
   macro avg     0.8776    0.8780    0.8778      7600
weighted avg     0.8776    0.8780    0.8778      7600


Results summary:
 {
  "TextCNN_test": {
    "loss": 0.30587584103147186,
    "acc": 0.9068421052631579,
    "f1m": 0.9067570188808819
  },
  "MLP_test": {
    "loss": 0.48491275931398076,
    "acc": 0.8780263157894737,
    "f1m": 0.8778008938537281
  },
  "TextCNN_params": 5938116,
  "MLP_params": 14301060
}

TextCNN acc=0.907 f1-macro=0.907 (5938116 params) | MLP acc=0.878 f1-macro=0.878 (14301060 params))
```

7. Ablations & Analysis

Ablations to be tried: (a) strip SE blocks; (b) strip depthwise separable conv (use non-separable conv); (c) turn off label smoothing; (d) no Adam, only Adam->SGD; (e) vary MAX_LEN. Inspect misclassifications qualitatively look at entity/keyword ambiguity and Long range dependencies.

8. Ethical Considerations

News included in the Dataset is written in English; possible biases are associated with the distributions of the sources. We do not create malicious substance and we make sure that it is reproducible through deterministic seeds. Information that can identify any person is not gathered other than the dataset.

9. Conclusion

TextCNN-SE offers a fast, compact baseline AG_NEWS. Refinements in SE attention and in training enhance large-scale robustness at lightweight computation. Future work: compare to subword tokenization, pretrained embeddings, or lightweight Transformers e.g. DistilBERT.

References

1. **TextCNN (baseline model)**

   Kim, Yoon. *Convolutional Neural Networks for Sentence Classification*. EMNLP 2014.
   Link: https://arxiv.org/abs/1408.5882

2. **Squeeze-and-Excitation (SE) Block**

   Hu, Jie, Li Shen, and Gang Sun. *Squeeze-and-Excitation Networks*. CVPR 2018.
   Link: https://arxiv.org/abs/1709.01507

3. **Depthwise-Separable Convolutions (MobileNets)**

   Howard, Andrew G., et al. *MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications*. 2017.
   Link: https://arxiv.org/abs/1704.04861

4. **Label Smoothing**

   Müller, Gabriel, et al. *When Does Label Smoothing Help?*. NeurIPS 2019.
   Link: https://arxiv.org/abs/1906.02629

5. **AdamW Optimizer**

   Loshchilov, Ilya, and Frank Hutter. *Decoupled Weight Decay Regularization (AdamW)*. ICLR 2019.
   Link: https://arxiv.org/abs/1711.05101

6. **SGDR (Cosine Annealing with Restarts)**

   Loshchilov, Ilya, and Frank Hutter. *SGDR: Stochastic Gradient Descent with Warm Restarts*. ICLR 2017.
   Link: https://arxiv.org/abs/1608.03983

7. **AG_NEWS Dataset**

   Provided via the Hugging Face datasets library.
   Link: https://huggingface.co/datasets/ag_news

8. **GitHub Repo**
   Link : https://github.com/Aishwariya16/AIDI1002_Final_Project.git