# Deep Neural Networks for Natural Language Processing (AI6127)

JUNG-JAE KIM

TUTORIAL 4: CONVOLUTIONAL NEURAL NETWORKS

# Question 1

- Run the iPython notebook (Section 5.3, “Document classification with CNN”) of the following site on your machine:
  - https://github.com/joosthub/PyTorchNLPBook (Repository of source codes explained in the book “Natural Language Processing with PyTorch” by Delip Rao and Brian McMahan)
  - It requires data file downloading and data pre-processing

# Hint 1

- Download ag_news dataset (see data/get-all-data.sh)
  - cd data; python download.py 1hjAZJJVyez-tjaUSwQyMBMVbW68Kgyzn ./ag_news/news.csv
- Download Glove word embeddings
  - http://nlp.stanford.edu/data/glove.6B.zip: unzip and place it under data/glove
- Run 5_3_Munging_AG_News notebook
- Run 5_3_Document_Classification_with_CNN notebook
  - Fix load_glove_from_file function in section “general utilities”
    - with open(glove_filepath, "r", encoding=‘utf8’) as fp:

# Question 2: Answer the following questions about the notebook

- How many layers of 1D CNN are used in the model?
  - Hint: See Section “The Model: NewsClassifier”, __init__ function
- What other layers are included in the model?
  - Hint: See Section “The Model: NewsClassifier”, __init__ and forward functions
- How many channels does each 1D CNN produce as output?
  - Hints:
    - See Section “The Model: NewsClassifier”, __init__ function
    - Sections “Initializations” and “Settings and some prep work”

# Question 2: Answer the following questions about the notebook

- What is difference between the padding introduced in the lecture slides and the padding introduced in the notebook?
  - Hints:
    - Section “The vectorizer”, vectorize function
    - Section “The dataset”, __init__ and __getitem__ functions
    - Section “The Vocabulary”, __init__ function

Recall

# A 1D convolution for text with padding = 1

| | | | | |
|---|---|---|---|---|
| ∅ | 0.0 | 0.0 | 0.0 | 0.0 |
| **tentative** | 0.2 | 0.1 | -0.3 | 0.4 |
| **deal** | 0.5 | 0.2 | -0.3 | -0.1 |
| **reached** | -0.2 | -0.3 | -0.2 | 0.4 |
| **to** | 0.3 | -0.3 | 0.1 | 0.1 |
| **keep** | 0.2 | -0.3 | 0.4 | 0.2 |
| **government** | 0.1 | 0.2 | -0.1 | -0.1 |
| **open** | -0.4 | -0.4 | 0.2 | 0.3 |
| ∅ | 0.0 | 0.0 | 0.0 | 0.0 |

| | |
|---|---|
| **∅,t,d** | -0.6 |
| **t,d,r** | -1.1 |
| **d,r,t** | -0.4 |
| **r,t,k** | -3.9 |
| **t,k,g** | -0.2 |
| **k,g,o** | 0.3 |
| **g,o, ∅** | -0.5 |

| | | | |
|---|---|---|---|
| 3 | 1 | 2 | -3 |
| -1 | 2 | 1 | -3 |
| 1 | 1 | -1 | 1 |

Apply a **filter** of size 3

Source: CS224n

# Question 2: Answer the following questions about the notebook

- What pooling method is used in the notebook?
    - Hint: See Section “The Model: NewsClassifier”, forward function
- Which regularisation method is used in the notebook?
    - Hint: See Section “The Model: NewsClassifier”, __init__ and forward functions

Recall

# Regularization: dropout

- Create masking vector *r* of Bernoulli random variables with probability *p* (a hyperparameter) of being 1
- Delete features during training: $y = \mathrm{softmax}\left(W^{(S)}(r \circ z) + b\right)$
  - E.g. randomly set 50% of the inputs to each neuron to 0, at each instance
- Reasoning: Prevents co-adaptation (overfitting to seeing specific feature constellations)
  - See the link below for details
- At test time, no dropout, scale down final vector by probability *p*: $\widehat{W}^{(S)} = pW^{(S)}$

https://jmlr.org/papers/volume15/srivastava14a.old/srivastava14a.pdf

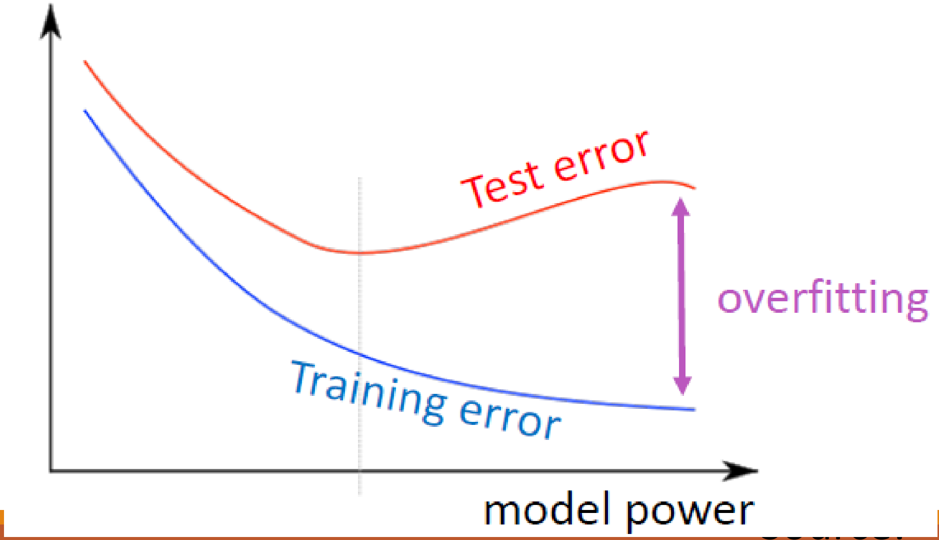
Source: CS224n

Recall

# Regularization: $l_2$ norm

- Recall: $\|\boldsymbol{v}\|_2 = \sqrt{v_1^2 + v_2^2 + v_3^2 + \cdots}$
- Constrain $l_2$ norms of weight vectors of each class (row in softmax weight $W^{(S)}$) to be upper-bounded by a fixed number *s* (also a hyperparameter)
  - If $\left\|W_c^{(S)}\right\|_2 > s$, then rescale it so that: $\left\|W_c^{(S)}\right\|_2 = s$
- Or, add regularization to loss function
  - Helps to prevent overfitting when we have a lot of features

$$J(\theta) = \frac{1}{N}\sum_{i=1}^{N} -\log\left(\frac{e^{f_{y_i}}}{\sum_{c=1}^{C} e^{f_c}}\right) \boxed{+\lambda\sum_{k}\theta_k^2}$$



https://jmlr.org/papers/volume15/srivastava14a.old/srivastava14a.pdf

Source: CS224n

# Hands-on

# Question 2: Answer the following questions about the notebook

- Question 2.1: How many layers of 1D CNN are used in the model?
- Hint: See Section “The Model: NewsClassifier”, __init__ function

# Answer 2.1

- Section “The Model: NewsClassifier”
- __init__ function

```
self.convnet = nn.Sequential(
        nn.Conv1d(in_channels=embedding_size,
             out_channels=num_channels, kernel_size=3),
        nn.ELU(),
        nn.Conv1d(in_channels=num_channels, out_channels=num_channels,
             kernel_size=3, stride=2),
        nn.ELU(),
        nn.Conv1d(in_channels=num_channels, out_channels=num_channels,
             kernel_size=3, stride=2),
        nn.ELU(),
        nn.Conv1d(in_channels=num_channels, out_channels=num_channels,
             kernel_size=3),
        nn.ELU()
     )
```

# Question 2.2: What other layers are included in the model?

- Hint: See Section “The Model: NewsClassifier”, __init__ and forward functions

# Answer 2.2

- __init__ function

```
self._dropout_p = dropout_p
self.fc1 = nn.Linear(num_channels, hidden_dim)
self.fc2 = nn.Linear(hidden_dim, num_classes)
```

- forward function

```
features = self.convnet(x_embedded)
# average and remove the extra dimension
remaining_size = features.size(dim=2)
features = F.avg_pool1d(features, remaining_size).squeeze(dim=2)
features = F.dropout(features, p=self._dropout_p)
# mlp classifier
intermediate_vector = F.relu(F.dropout(self.fc1(features), p=self._dropout_p))
prediction_vector = self.fc2(intermediate_vector)
if apply_softmax:
        prediction_vector = F.softmax(prediction_vector, dim=1)
```

# Question 2.3: How many channels does each 1D CNN produce as output?

- Hints:
  - See Section “The Model: NewsClassifier”, __init__ function
  - Section “Initializations”
  - Section “Settings and some prep work”

# Answer 2.3

- Section “The Model: NewsClassifier”
- __init__ function

```
self.convnet = nn.Sequential(
        nn.Conv1d(in_channels=embedding_size,
             out_channels=num_channels,
             kernel_size=3),
        nn.ELU(),
        nn.Conv1d(in_channels=num_channels,
             out_channels=num_channels,
             kernel_size=3, stride=2),
        nn.ELU(),
        nn.Conv1d(in_channels=num_channels,
             out_channels=num_channels,
             kernel_size=3, stride=2),
        nn.ELU(),
        nn.Conv1d(in_channels=num_channels,
             out_channels=num_channels,
             kernel_size=3),
        nn.ELU()
     )
```

- Section “Initializations”
  - classifier = NewsClassifier(…, num_channels=args.num_channels, …)
- Section “Settings and some prep work”
  - args = Namespace(…, num_channels=100, …)

# Question 2.4: What is difference between the padding introduced in the lecture slides and the padding introduced in the notebook?

- Hints:
  - Section “The vectorizer”, vectorize function
  - Section “The dataset”, __init__ and __getitem__ functions
  - Section “The Vocabulary”, __init__ function

# Answer 2.4

- Section “The vectorizer”, vectorize function
  - indices = [self.title_vocab.begin_seq_index]
  - indices.extend(self.title_vocab.lookup_token(token) for token in title.split(" "))
  - indices.append(self.title_vocab.end_seq_index)
  - if vector_length < 0: vector_length = len(indices)
  - out_vector = np.zeros(vector_length, dtype=np.int64)
  - out_vector[:len(indices)] = indices
  - out_vector[len(indices):] = self.title_vocab.mask_index
- Section “The dataset”
  - __init__ function
    - measure_len = lambda context: len(context.split(" "))
    - self._max_seq_length = max(map(measure_len, news_df.title)) + 2
  - __getitem__ function
    - title_vector = self._vectorizer.vectorize(row.title, self._max_seq_length)

# Answer 2.4

- Section “The Vocabulary”
  - def __init__(self, token_to_idx=None, unk_token="<UNK>", mask_token="<MASK>", begin_seq_token="<BEGIN>", end_seq_token="<END>"):
    - self.mask_index = self.add_token(self._mask_token)
    - self.unk_index = self.add_token(self._unk_token)
    - self.begin_seq_index = self.add_token(self._begin_seq_token)
    - self.end_seq_index = self.add_token(self._end_seq_token)

# Question 2.5: What pooling method is used in the notebook?

- Hint: See Section “The Model: NewsClassifier”, forward function

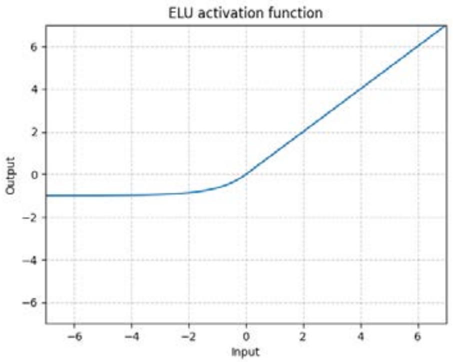
# Answer 2.5

- features = F.avg_pool1d(features, remaining_size).squeeze(dim=2)

# Question 2.6: Which regularisation method is used in the notebook?

- Hint: See Section “The Model: NewsClassifier”, __init__ and forward functions

# Answer 2.6



- Section “The Model: NewsClassifier”
- __init__ function

```
self.convnet = nn.Sequential(
        nn.Conv1d(in_channels=embedding_size,
             out_channels=num_channels,
             kernel_size=3),
        nn.ELU(),
        nn.Conv1d(in_channels=num_channels,
             out_channels=num_channels,
             kernel_size=3, stride=2),
        nn.ELU(),
        nn.Conv1d(in_channels=num_channels,
             out_channels=num_channels,
             kernel_size=3, stride=2),
        nn.ELU(),
        nn.Conv1d(in_channels=num_channels,
             out_channels=num_channels,
             kernel_size=3),
        nn.ELU()
    )
```

- forward function
  - features = F.dropout(features, p=self._dropout_p)
  - intermediate_vector = F.relu(F.dropout(self.fc1(features), p=self._dropout_p))

# Answer 2: Summary

- 4 layers of 1D CNN
- 4 non-linear layers; 2 fully connected layers; 1 avg pooling layer
- 100 output channels for each 1D CNN
- padding with special token (mask_index) is used in notebook, while padding of 0.0 is used in lecture slides
- average pooling method
- dropout regularization