Popular Transformers

Danna Gurari
University of Colorado Boulder
Spring 2024

https://dannagurari.colorado.edu/course/neural-networks-and-deep-learning-spring-2024/
Review

• Last lecture:
  • Transformer overview
  • Self-attention
  • Multi-head attention
  • Common transformer ingredients
  • Pioneering transformer: machine translation

• Assignments (Canvas):
  • Lab assignment 3 due earlier today
  • Problem set 4 (last one!) due in one week
  • Final project after spring break (find partner, including at 5-6pm zoom meeting)

• Questions?
Today’s Topics

• Explosion of transformers

• GPT

• BERT

• ViT

• Programming tutorial
Today’s Topics

• Explosion of transformers

• GPT

• BERT

• ViT

• Programming tutorial
Explosion of Transformers in Society

https://www.pinterest.com/pin/521784306804400819/
Today’s Focus: Methods that Perform Pretraining and then Fine-tuning

Transformers can provide better embeddings for downstream tasks since they capture context (i.e., unlike context-free embeddings such as word2vec, the word embedding is different for a word used in different contexts... e.g., “I arrived at the bank after crossing the river/street”)

Today’s Focus: Historical Context


- First programmable machine
- Turing test
- AI
- Perceptron
- Machine learning
- Backpropagation for CNNs
- MNIST, LeNet
- Wave 3: rise of “deep learning”
- Neocognitron
- Neural networks with effective learning strategy
- GPT, BERT
- ViT
Today’s Topics

• Explosion of transformers

• GPT

• BERT

• ViT

• Programming tutorial
GPT: Generative Pre-Training

Pre-Training
- Large unlabelled datasets (e.g. Wikipedia, BookCorpus)
- Self-supervised training (hours to days)

Fine-Tuning
- Smaller labelled datasets (SQuAD, MNLI/CMNLI, Similarity)
- Task-specific fine tuning (minutes to hours)

Inference

Task: Predict Next Word Given Previous Ones

e.g.,

1. Background music from a __________

2. Many people danced around the __________

3. I practiced for many years to learn how to play the __________
Architecture: Decoder from Pioneering Transformer

Removes encoder and corresponding decoder sub-layer thereby ignoring translation of an input sentence!
Architecture: Autoregressive Model

“Autoregressive” because previously generated tokens are used to predict subsequent tokens

https://jalammar.github.io/illustrated-transformer/
Architecture: Simplified Figure

Architecture: **Input & Output**

- **Embedding with Time Signal**
- **Positional Encoding**
- **Embeddings**

**INPUT**
- Je
- suis
- étudiant

512 tokens used

Softmax layer determines next predicted word at each time step

http://jalammar.github.io/illustrated-transformer/

Architecture: Masked Attention

Limit each word’s new representation to only reflect earlier words (mimics inference time when only previous tokens can be seen):

- e.g., at <sos> token, no previous inputs
- e.g., at 3rd token, have two previous inputs

https://stackoverflow.com/questions/64799622/how-is-the-gpts-masked-self-attention-is-utilized-on-fine-tuning-inference

Architecture: **Masked Attention**

Limit each word’s new representation to only reflect earlier words (mimics inference time when only previous tokens can be seen):

Masked out scores are represented as negative infinity so the softmax result (i.e., attention weight) returns 0.
Training

- Dataset: 800M words from BooksCorpus (>7,000 books)
- Optimizer: Adam
- Training duration: 100 epochs
Implementation Details

Activation function: Gaussian error linear unit (GELU)

\[
g(x) = 0.5x \left(1 + \frac{2}{\sqrt{\pi}} \int_0^{x/\sqrt{2}} e^{-t^2} dt\right)
\]

https://datascience.stackexchange.com/questions/49522/what-is-gelu-activation
Avoid out of vocabulary tokens with subword tokenization (byte pair encoding):
1. Identify all tokens in the training data with their frequency
2. Define vocabulary size; e.g., 14
3. Add all characters in the tokenized input to the vocabulary; e.g.,

<table>
<thead>
<tr>
<th>Character sequence</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cost</td>
<td>2</td>
</tr>
<tr>
<td>best</td>
<td>2</td>
</tr>
<tr>
<td>menu</td>
<td>1</td>
</tr>
<tr>
<td>men</td>
<td>1</td>
</tr>
<tr>
<td>camel</td>
<td>1</td>
</tr>
</tbody>
</table>
Avoid out of vocabulary tokens with subword tokenization (byte pair encoding):
1. Identify all tokens in the training data with their frequency
2. Define vocabulary size; e.g., 14
3. Add all characters in the tokenized input to the vocabulary; e.g.,
4. Until vocabulary is filled, add merged highest frequency symbol pairs

<table>
<thead>
<tr>
<th>Character sequence</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cost</td>
<td>2</td>
</tr>
<tr>
<td>best</td>
<td>2</td>
</tr>
<tr>
<td>menu</td>
<td>1</td>
</tr>
<tr>
<td>men</td>
<td>1</td>
</tr>
<tr>
<td>camel</td>
<td>1</td>
</tr>
</tbody>
</table>

Vocabulary: a, b, c, e, l, m, n, o, s, t, u

e.g., What are the highest frequency symbol pairs?

Avoid out of vocabulary tokens with subword tokenization (byte pair encoding):
1. Identify all tokens in the training data with their frequency
2. Define vocabulary size; e.g., 14
3. Add all characters in the tokenized input to the vocabulary; e.g.,
4. Until vocabulary is filled, add merged highest frequency symbol pairs

### Example

<table>
<thead>
<tr>
<th>Character sequence</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cost</td>
<td>2</td>
</tr>
<tr>
<td>best</td>
<td>2</td>
</tr>
<tr>
<td>menu</td>
<td>1</td>
</tr>
<tr>
<td>men</td>
<td>1</td>
</tr>
<tr>
<td>camel</td>
<td>1</td>
</tr>
</tbody>
</table>

**Vocabulary**

a, b, c, e, l, m, n, o, s, t, u, st

---

e.g., What are the highest frequency symbol pairs?
Avoid out of vocabulary tokens with subword tokenization (byte pair encoding):
1. Identify all tokens in the training data with their frequency
2. Define vocabulary size; e.g., 14
3. Add all characters in the tokenized input to the vocabulary; e.g.,
4. Until vocabulary is filled, add merged highest frequency symbol pairs

<table>
<thead>
<tr>
<th>Character sequence</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cost</td>
<td>2</td>
</tr>
<tr>
<td>best</td>
<td>2</td>
</tr>
<tr>
<td>menu</td>
<td>1</td>
</tr>
<tr>
<td>men</td>
<td>1</td>
</tr>
<tr>
<td>camel</td>
<td>1</td>
</tr>
</tbody>
</table>

**Vocabulary**
a, b, c, e, i, m, n, o,
s, t, u, st, me, men

e.g., What are the highest frequency symbol pairs?
Avoid out of vocabulary tokens with subword tokenization (byte pair encoding):
- 40,000 merges used
GPT: Generative Pre-Training

Pre-Training
- Large unlabelled datasets (e.g. Wikipedia, BookCorpus)
- Self-supervised training (hours to days)

Fine-Tuning
- Smaller labelled datasets (SQuAD, MNLI/CMNLI, Similarity)
- Task-specific fine tuning (minutes to hours)

Inference

Fine-Tuning to Target Task (Softmax Output); i.e.,

Experimental Findings

Achieved the best performance on 9 NLP dataset challenges
GPT: Generative Pre-Training

**Pre-Training**
- Large unlabelled datasets (e.g. Wikipedia, BookCorpus)
- Self-supervised training (hours to days)

**Fine-Tuning**
- Smaller labelled datasets (SQuAD, MNLI/CMNLI, Similarity)
- Task-specific fine tuning (minutes to hours)

**Inference**

Today’s Topics

• Explosion of transformers

• GPT

• BERT

• ViT

• Programming tutorial
Motivation: Choose a Pretraining Task That Is Not Unidirectional

GPT’s prediction of the next word given previous ones is unidirectional (left-to-right)

1. Background music from a _______
2. Many people danced around the _______
3. I practiced for many years to learn how to play the _______
BERT: **Bidirectional Encoder Representation from Transformer**

**Pre-Training**
- Large unlabelled datasets (e.g. Wikipedia, BookCorpus)
- Self-supervised training (hours to days)

**Fine-Tuning**
- Smaller labelled datasets (SQuAD, MNLI/CMNLI, Similarity)
- Task-specific fine tuning (minutes to hours)

**Inference**

Two Tasks

1. Predict masked token (key contribution)

2. Predict if one sentence follows a second sentence (augments understanding of how sentences relate)
Architecture: Encoder from Pioneering Transformer

Architecture: Variants

- $L =$ number of stacked encoders
- $H =$ number of hidden units in feedforward layer

<table>
<thead>
<tr>
<th>$L$</th>
<th>$H=128$</th>
<th>$H=256$</th>
<th>$H=512$</th>
<th>$H=768$</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>2/128 (BERT-tiny)</td>
<td>2/256</td>
<td>2/512</td>
<td>2/768</td>
</tr>
<tr>
<td>4</td>
<td>4/128</td>
<td>4/256 (BERT-mini)</td>
<td>4/512 (BERT-small)</td>
<td>4/768</td>
</tr>
<tr>
<td>6</td>
<td>6/128</td>
<td>6/256</td>
<td>6/512</td>
<td>6/768</td>
</tr>
<tr>
<td>8</td>
<td>8/128</td>
<td>8/256</td>
<td>8/512 (BERT-medium)</td>
<td>8/768</td>
</tr>
<tr>
<td>10</td>
<td>10/128</td>
<td>10/256</td>
<td>10/512</td>
<td>10/768</td>
</tr>
<tr>
<td>12</td>
<td>12/128</td>
<td>12/256</td>
<td>12/512</td>
<td>12/768 (BERT-base)</td>
</tr>
</tbody>
</table>

BERT-large ($H = 1024$)

Input sentence

Encoder 1

Encoder 2

Encoder 3

Encoder 4

Representation

Architecture: BERT-Base (Matches Size of GPT)
Architecture: Input

Two input sentences with [CLS] at the start and [SEP] between sentences

Input: [CLS] Paris is a beautiful [MASK] [SEP] I love Paris [SEP]
Input is addition of a segment embedding to the token and position embeddings (helps differentiate which tokens belong to which sentence)
New representation of each input that accounts for context.

Input: [CLS] Paris is a beautiful [MASK] [SEP] I love Paris [SEP]

- **Token embeddings**
  - $E_{[CLS]}$, $E_{Paris}$, $E_{is}$, $E_{a}$, $E_{beautiful}$, $E_{city}$, $E_{[SEP]}$, $E_{I}$, $E_{love}$, $E_{Paris}$, $E_{[SEP]}$
  - $E_0$, $E_1$, $E_2$, $E_3$, $E_4$, $E_5$, $E_6$, $E_7$, $E_8$, $E_9$, $E_{10}$

- **Segment embeddings**
  - $E_A$, $E_A$, $E_A$, $E_A$, $E_A$, $E_A$, $E_A$, $E_B$, $E_B$, $E_B$, $E_B$

- **Position embeddings**
  - $E_0$, $E_1$, $E_2$, $E_3$, $E_4$, $E_5$, $E_6$, $E_7$, $E_8$, $E_9$, $E_{10}$

Diagram shows Encoder 1, Encoder 2, Encoder 3, and Encoder 12.
Architecture: Predicting Masked Token Task

15% of random tokens from sequence masked
- 80% use [MASK]
- 10% use a random token
- 10% use original token
Multiple masking options encourage the model to pay attention to each token separately

Probability distribution over output vocabulary

Feed forward network + softmax

Encoder 12
⋯
Encoder 3
Encoder 2
Encoder 1

[CLS] Paris is a beautiful [MASK] [SEP] I love Paris [SEP]
Architecture: Predict if Next Sentence Task

Predict with token representation that aggregates representation of all tokens in both sentences.

50% of 2nd sentences are the original next sentence and the rest are random.
Training

- Dataset: 2,500M words in Wikipedia + 800M words in BooksCorpus used for GPT

- Optimizer: Adam

- Training loss: sums over losses from predicting masked words and if next sentence
Implementation Details: Mimics GPT

- Gaussian error linear unit (GELU) used as activation function in feedforward layers

\[ g(x) = 0.5x \left( 1 + \frac{2}{\sqrt{\pi}} \int_0^{\frac{x}{\sqrt{2}}} e^{-t^2} dt \right) \]

https://datascience.stackexchange.com/questions/49522/what-is-gelu-activation

- Avoids out of vocabulary tokens by using subword tokenization, with a different variant called WordPiece Tokenization
BERT: Bidirectional Encoder Representation from Transformer

Pre-Training
- Large unlabelled datasets (e.g. Wikipedia, BookCorpus)
- Self-supervised training (hours to days)

Fine-Tuning
- Smaller labelled datasets (SQuAD, MNLI/CMNLI, Similarity)
- Task-specific fine tuning (minutes to hours)

Inference

# Fine-Tuning for Natural Language Inference

<table>
<thead>
<tr>
<th>Premise</th>
<th>Hypothesis</th>
<th>Label</th>
</tr>
</thead>
<tbody>
<tr>
<td>He is playing</td>
<td>He is sleeping</td>
<td>Contradiction</td>
</tr>
<tr>
<td>A soccer game with multiple males playing</td>
<td>Some men are playing sport</td>
<td>Entailment</td>
</tr>
<tr>
<td>An older and a younger man smiling</td>
<td>Two men are smiling at the dogs playing on the floor</td>
<td>Neutral</td>
</tr>
</tbody>
</table>
Fine-Tuning for Natural Language Inference

Pre-trained
BERT

Contradiction
Entailment
Neutral

Feed forward network + softmax
To find indices of the start and end words in the paragraph, two vector representations are learned that lead to the approximate softmax output when computing the dot product with each token.
Fine-Tuning for Named Entity Recognition

Each token’s new representation is passed to a classifier
Experimental Findings

Achieved the best performance on 11 NLP dataset challenges

Experimental Findings

<table>
<thead>
<tr>
<th>Tasks</th>
<th>MNLI-m (Acc)</th>
<th>QNLI (Acc)</th>
<th>MRPC (Acc)</th>
<th>SST-2 (Acc)</th>
<th>SQuAD (F1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>BERT\textsubscript{BASE}</td>
<td>84.4</td>
<td>88.4</td>
<td>86.7</td>
<td>92.7</td>
<td>88.5</td>
</tr>
<tr>
<td>No NSP</td>
<td>83.9</td>
<td>84.9</td>
<td>86.5</td>
<td>92.6</td>
<td>87.9</td>
</tr>
<tr>
<td>LTR &amp; No NSP</td>
<td>82.1</td>
<td>84.3</td>
<td>77.5</td>
<td>92.1</td>
<td>77.8</td>
</tr>
<tr>
<td>+ BiLSTM</td>
<td>82.1</td>
<td>84.1</td>
<td>75.7</td>
<td>91.6</td>
<td>84.9</td>
</tr>
</tbody>
</table>

Next sentence prediction (NSP) supports slight improvements
Experimental Findings

<table>
<thead>
<tr>
<th>Tasks</th>
<th>MNLI-m (Acc)</th>
<th>QNLI (Acc)</th>
<th>MRPC (Acc)</th>
<th>SST-2 (Acc)</th>
<th>SQuAD (F1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>BERT_BASE</td>
<td>84.4</td>
<td>88.4</td>
<td>86.7</td>
<td>92.7</td>
<td>88.5</td>
</tr>
<tr>
<td>No NSP</td>
<td>83.9</td>
<td>84.9</td>
<td>86.5</td>
<td>92.6</td>
<td>87.9</td>
</tr>
<tr>
<td>LTR &amp; No NSP</td>
<td>82.1</td>
<td>84.3</td>
<td>77.5</td>
<td>92.1</td>
<td>77.8</td>
</tr>
<tr>
<td>+ BiLSTM</td>
<td>82.1</td>
<td>84.1</td>
<td>75.7</td>
<td>91.6</td>
<td>84.9</td>
</tr>
</tbody>
</table>

We observe a performance boost when using bidirectional pretraining instead of unidirectional pretraining (LTR)

BERT: Bidirectional Encoder Representation from Transformer

Pre-Training
- Large unlabelled datasets (e.g. Wikipedia, BookCorpus)
- Self-supervised training (hours to days)

Fine-Tuning
- Smaller labelled datasets (SQuAD, MNLI/CMNLI, Similarity)
- Task-specific fine tuning (minutes to hours)

Inference

Today’s Topics

• Explosion of transformers

• GPT

• BERT

• ViT

• Programming tutorial
ViT: Vision Transformer

Pre-Training
- Large, labelled datasets (e.g. Wikipedia, BookCorpus)
- Self-supervised training (hours to days)

Fine-Tuning
- Smaller labelled datasets (SQuAD, MNLI/CMNLI, Similarity)
- Task-specific fine tuning (minutes to hours)

Inference

Motivation: Transformers for Image Classification (Repurpose BERT)

Dosovitskiy et al. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. ICLR 2021.
ViT: Vision Transformer

Pre-Training
- Large unlabeled datasets (e.g., Wikipedia, BookCorpus)
- Self-supervised training (hours to days)

Fine-Tuning
- Smaller labeled datasets (SQuAD, MNLI/CMNLI, Similarity)
- Task-specific fine tuning (minutes to hours)

Inference

Architecture

Dosovitskiy et al. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. ICLR 2021.
Architecture: BERT

Dosovitskiy et al. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. ICLR 2021.
Dosovitskiy et al. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. ICLR 2021.
Architecture: Input (Patches Instead of Pixels)

Why not use the raw pixels as input?

Dosovitskiy et al. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. ICLR 2021.
Architecture: Input CLS Token

[CLS] token represents entire image

Dosovitskiy et al. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. ICLR 2021.
Architecture: Training

- Dataset: JFT with 303M labeled images (proprietary Google dataset)
- Task: classification loss (supervised)
- Optimizer: Adam

* Note: research also is exploring how smaller training datasets can be effective; e.g., data efficient image transformers (DeiT) from “Training data-efficient image transformers & distillation through attention”
ViT: **Vision Transformer**

### Pre-Training
- Large, unlabelled datasets (e.g. Wikipedia, BookCorpus)
- Semi-supervised training (hours to days)

### Fine-Tuning
- Smaller labelled datasets (SQuAD, MNLI/CMNLI, Similarity)
- Task-specific fine tuning (minutes to hours)

### Inference

Fine-Tuning for Other Image Classification Tasks

MLP replaced with a single linear layer when fine-tuning to new classification categories
Experimental Findings

Achieved strong results on five image classification datasets

Transformers vs CNNs

Self-attention: each layer has a global receptive field

Convolutional layers: deeper layers have increasingly more global receptive fields

https://towardsdatascience.com/self-attention-5b95ea164f61

https://www.deeplearningbook.org/contents/convnets.html
Today’s Topics

• Explosion of transformers
• GPT
• BERT
• ViT
• Programming tutorial
Today’s Topics

• Explosion of transformers
• GPT
• BERT
• ViT
• Programming tutorial