Topics:
• Masked Language Models (dropbox M3L12)
• Embeddings (dropbox M3L13)
• Reinforcement Learning introduction

CS 4644-DL / 7643-A
ZSOLT KIRA
• **Assignment 4 out**
  • Due **April 4th 11:59pm EST (grace April 6th)**
  • Do not submit first version last-minute on 6th! 
    • Please submit *something* by deadline (Apr 4th) to avoid last-minute hiccups and zero!

• **Projects**
  • Project due **May 1st 11:59pm EST**

• **Outline of rest of course:**
  • Today we start (deep) reinforcement learning
  • Guest lectures/other topics (e.g. self-supervised learning)
  • Generative models (VAEs / GANs)
Sequences in Input or Output?

- It’s one to one.
- Input: No sequence
  Output: No sequence
  Example: "standard"

- Input: No sequence
  Output: Sequence
  Example: Im2Caption

- Input: Sequence
  Output: No sequence
  Example: sentence classification

- Input: Sequence
  Output: Sequence
  Example: machine translation, video classification, video captioning, open-ended question answering
The Space of Architectures

- Fully Connected Neural Networks
- Convolutional Neural Networks
- Recurrent Neural Networks
- Attention-Based Networks

Graph-Based Networks
Recall: Transformers

Transformer Block

Multi-Layered

Encoder/Decoder

Georgia Tech
Masked Language Models
Jean Maillard

Jean Maillard is a Research Scientist on the Language And Translation Technologies Team (LATTE) at Facebook AI. His research interests within NLP include word- and sentence-level semantics, structured prediction, and low-resource languages. Prior to joining Facebook in 2019, he was a doctoral student with the NLP group at the University of Cambridge, where he researched compositional semantic methods. He received his BSc in Theoretical Physics from Imperial College London.
Recall: language models estimate the probability of sequences of words:

\[ p(s) = p(w_1, w_2, \ldots, w_n) \]

Masked language modeling is a related pre-training task – an auxiliary task, different from the final task we’re really interested in, but which can help us achieve better performance by finding good initial parameters for the model.

By pre-training on masked language modeling before training on our final task, it is usually possible to obtain higher performance than by simply training on the final task.
Masked Language Models

take a seat, have a drink
<s> <mask> a seat <mask> have a <mask> </s>
Masked Language Models
Masked Language Models
Masked Language Models
Token-level Tasks
Token-level Tasks
Sentence-level Tasks
Cross-lingual Masked Language Modeling
<s> I am <mask> <sep> J' <mask> faim </s>
Cross-lingual Masked Language Modeling
Cross-lingual Task: Natural Language Inference
Cross-lingual Task: Natural Language Inference
Model Size in Perspective

- **RoBERTa (2019)**
- **T5 (2019)**
- **BERT (2018)**
- **ELMo (2018)**
- **CBOW (2013)**

Types of model:
- Shallow network
- RNN
- Transformer

Performance (GLUE) vs. Number of parameters (millions)
AN IMAGE IS WORTH 16x16 WORDS: TRANSFORMERS FOR IMAGE RECOGNITION AT SCALE

Alexey Dosovitskiy*, Lucas Beyer*, Alexander Kolesnikov*, Dirk Weissenborn*, Xiaohua Zhai†, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, Neil Houlsby*†

*equal technical contribution, †equal advising
Google Research, Brain Team
{adosovitskiy, neilhoulsby}@google.com

ABSTRACT

While the Transformer architecture has become the de-facto standard for natural language processing tasks, its applications to computer vision remain limited. In vision, attention is either applied in conjunction with convolutional networks, or used to replace certain components of convolutional networks while keeping their overall structure in place. We show that this reliance on CNNs is not necessary and a pure transformer applied directly to sequences of image patches can perform very well on image classification tasks. When pre-trained on large amounts of data and transferred to multiple mid-sized or small image recognition benchmarks (ImageNet, CIFAR-100, VTAB, etc.), Vision Transformer (ViT) attains excellent results compared to state-of-the-art convolutional networks while requiring substantially fewer computational resources to train.
Vision Transformer (ViT)

Transformer Encoder

MLP
Norm
Multi-Head Attention
Norm
Embedded Patches

Transformer Encoder

Linear Projection of Flattened Patches

Patch + Position Embedding
* Extra learnable [class] embedding

Class
Bird
Ball
Car...
<table>
<thead>
<tr>
<th>Model</th>
<th>Layers</th>
<th>Hidden size ( D )</th>
<th>MLP size</th>
<th>Heads</th>
<th>Params</th>
</tr>
</thead>
<tbody>
<tr>
<td>ViT-Base</td>
<td>12</td>
<td>768</td>
<td>3072</td>
<td>12</td>
<td>86M</td>
</tr>
<tr>
<td>ViT-Large</td>
<td>24</td>
<td>1024</td>
<td>4096</td>
<td>16</td>
<td>307M</td>
</tr>
<tr>
<td>ViT-Huge</td>
<td>32</td>
<td>1280</td>
<td>5120</td>
<td>16</td>
<td>632M</td>
</tr>
</tbody>
</table>

Table 1: Details of Vision Transformer model variants.

<table>
<thead>
<tr>
<th></th>
<th>Ours-JFT (ViT-H/14)</th>
<th>Ours-JFT (ViT-L/16)</th>
<th>Ours-I21K (ViT-L/16)</th>
<th>BiT-L (ResNet152x4)</th>
<th>Noisy Student (EfficientNet-L2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ImageNet</td>
<td>88.55 ± 0.04</td>
<td>87.76 ± 0.03</td>
<td>85.30 ± 0.02</td>
<td>87.54 ± 0.02</td>
<td>88.4/88.5*</td>
</tr>
<tr>
<td>ImageNet ReaL</td>
<td>90.72 ± 0.05</td>
<td>90.54 ± 0.03</td>
<td>88.62 ± 0.05</td>
<td>90.54</td>
<td>90.55</td>
</tr>
<tr>
<td>CIFAR-10</td>
<td>99.50 ± 0.06</td>
<td>99.42 ± 0.03</td>
<td>99.15 ± 0.03</td>
<td>99.37 ± 0.06</td>
<td>–</td>
</tr>
<tr>
<td>CIFAR-100</td>
<td>94.55 ± 0.04</td>
<td>93.90 ± 0.05</td>
<td>93.25 ± 0.05</td>
<td>93.51 ± 0.08</td>
<td>–</td>
</tr>
<tr>
<td>Oxford-IIIT Pets</td>
<td>97.56 ± 0.03</td>
<td>97.32 ± 0.11</td>
<td>94.67 ± 0.15</td>
<td>96.62 ± 0.23</td>
<td>–</td>
</tr>
<tr>
<td>Oxford Flowers-102</td>
<td>99.68 ± 0.02</td>
<td>99.74 ± 0.00</td>
<td>99.61 ± 0.02</td>
<td>99.63 ± 0.03</td>
<td>–</td>
</tr>
<tr>
<td>VTAB (19 tasks)</td>
<td>77.63 ± 0.23</td>
<td>76.28 ± 0.46</td>
<td>72.72 ± 0.21</td>
<td>76.29 ± 1.70</td>
<td>–</td>
</tr>
<tr>
<td>TPUv3-core-days</td>
<td>2.5k</td>
<td>0.68k</td>
<td>0.23k</td>
<td>9.9k</td>
<td>12.3k</td>
</tr>
</tbody>
</table>
When trained on mid-sized datasets such as ImageNet, such models yield modest accuracies of a few percentage points below ResNets of comparable size. This seemingly discouraging outcome maybe expected: Transformers lack some of the inductive biases inherent to CNNs, such as translation equivariance and locality, and therefore do not generalize well when trained on insufficient amounts of data.

However, the picture changes if the models are trained on larger datasets (14M-300M images). We find that large scale training trumps inductive bias.

Figure 3: Transfer to ImageNet. While large ViT models perform worse than BiT ResNets (shaded area) when pre-trained on small datasets, they shine when pre-trained on larger datasets. Similarly, larger ViT variants overtake smaller ones as the dataset grows.
Swin Transformer: Hierarchical Vision Transformer using Shifted Windows
Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, Baining Guo

https://paperswithcode.com/sota/instance-segmentation-on-coco

Swin Transformers
Summary

• “Attention” models outperform recurrent models and convolutional models for sequence processing. They allow long range interactions.
• These models do best with LOTS of training data
• Surprisingly, they seem to outperform convolutional networks for image processing tasks. Again, long range interactions might be more important than we realized.
• Naïve attention mechanisms have quadratic complexity with the number of input tokens, but there are often workarounds for this.
Knowledge Distillation to Reduce Model Sizes
Knowledge Distillation to Reduce Model Sizes

Pretrained teacher model → Soft predictions

Student model → Soft predictions

Input text

Target

the <mask> licked its fur and howled

[Bar chart with categories: wolf, singing, dog, fox, pineapple]
cross-entropy

\[ H(p^*, p) = - \sum_{x \in X} p^*(x) \log p(x) \]

reference distribution

\[ \mathcal{L}_{\text{dist}} = H(t, s) = - \sum_i t_i \log s_i \quad \text{or} \quad D_{KL}(t||s) \]

\[ \mathcal{L}_{\text{student}} = H(y, s) = - \sum_i y_i \log s_i \]

\[ \mathcal{L} = \alpha \mathcal{L}_{\text{dist}} + \beta \mathcal{L}_{\text{student}} \]


Ledell Wu

Ledell Wu is a research engineer at Facebook AI Research. Ledell joined Facebook in 2013 after graduating from University of Toronto. She worked on Newsfeed ranking as a machine learning engineer. After joining Facebook AI, Ledell worked on general purpose and large-scale embedding systems. She collaborated with teams including page recommendations, video recommendations, ads interest suggestion, people search and feed integrity, to use embeddings to better serve products. She is one of the main contributors in open source projects including StarSpace (general purpose embedding system), PyTorch Big-Graph (large-scale graph embedding system) and BLINK (entity linking). Ledell also studies fairness and biases in machine learning models.
Embeddings

- Word Embeddings
- Graph Embeddings
- Applications, world2vec
- Additional Topics
Mapping Objects to Vectors through a trainable function

[0.4, -1.3, 2.5, -0.7, …]

[0.2, -2.1, 0.4, -0.5, …]

“The neighbors’ dog was a Samoyed, which looks a lot like a Siberian husky”

Slide Credit: Yann LeCun
Introduction to Embeddings

Joyeux Anniversaire!

#IamOld

Happy Birthday 😊 😊

#vannsplanes

Wow! Checkout this vintage ramjet.

Watched John Coltrane tribute concert last Sun.
(Big) Graph Data is Everywhere

**Knowledge Graphs**
Standard domain for studying graph embeddings (*Freebase, …*)

**Recommender Systems**
Deals with graph-like data, but supervised

**Social Graphs**
Predict attributes based on homophily or structural similarity (*Twitter, Yelp, …*)

---


*Slide Credit: Adam Lerer*
**Graph Embedding & Matrix Completion**

<table>
<thead>
<tr>
<th></th>
<th>item1</th>
<th>item2</th>
<th></th>
<th>itemN</th>
</tr>
</thead>
<tbody>
<tr>
<td>person1</td>
<td>-</td>
<td>+</td>
<td></td>
<td>+</td>
</tr>
<tr>
<td>person2</td>
<td>+</td>
<td>?</td>
<td></td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>personP</td>
<td>+</td>
<td>-</td>
<td></td>
<td>?</td>
</tr>
</tbody>
</table>

- Relations between items (and people)
- Items in \{people, movies, page, articles, products, word sequences\}...
- Predict if someone will like an item, if a word will follow a word sequence

*Slide Credit: Yann LeCun*
**Embedding**: A learned map from entities to vectors of numbers that encodes similarity

- Word embeddings: word ⟷ vector
- Graph embeddings: node ⟷ vector

**Graph Embedding**: Optimize the objective that connected nodes have more similar embeddings than unconnected nodes via gradient descent.
Why Graph Embeddings?

Graph embeddings are a form of unsupervised learning on graphs.

- **Task-agnostic** entity representations
- Features are useful on downstream tasks without much data
- Nearest neighbors are semantically meaningful

*A multi-relation graph*
Margin loss between the score for an edge $f(e)$ and a negative sampled edge $f(e')$

$$
\mathcal{L} = \sum_{e \in S} \sum_{e' \in S'_e} \max(f(e) - f(e') + \lambda, 0))
$$

The score for an edge is a similarity (e.g. dot product) between the source embedding and a transformed version of the destination embedding, e.g.

$$
f(e) = \cos(\theta_s, \theta_r + \theta_d)
$$

Negative samples are constructed by taking a real edge and replacing the source or destination with a random node.

$$
S'_e = \{(s', r, d) | s' \in V \} \cup \{(s, r, d' | d' \in V \}
$$
Graph Embeddings

Margin loss between the score for an edge $f(e)$ and a negative sampled edge $f(e')$

$$L = \sum_{e \in E} \sum_{e' \in S'_e} \max(f(e) - f(e') + \lambda, 0))$$

The score for an edge is a similarity (e.g. dot product) between the source embedding and a transformed version of the destination embedding, e.g.

$$f(e) = \cos(\theta_s, \theta_r + \theta_d)$$

Negative samples are constructed by taking a real edge and replacing the source or destination with a random node.

$$S'_e = \{ (s', r, d') | s' \in V \} \cup \{ (s, r, d' | d' \in V \}$$

Slide Credit: Adam Lerer
Margin loss between the score for an edge $f(e)$ and a negative sampled edge $f(e')$

$$L = \sum_{e \in E} \sum_{e' \in S'_e} \max (f(e) - f(e') + \lambda, 0)$$

The score for an edge is a similarity (e.g. dot product) between the source embedding and a transformed version of the destination embedding, e.g.

$$f(e) = \cos(\theta_s, \theta_r + \theta_d)$$

Negative samples are constructed by taking a real edge and replacing the source or destination with a random node.

$$S'_e = \{(s', r, d') | s' \in V \} \cup \{(s, r, d' | d' \in V \}$$

Slide Credit: Adam Lerer
Multiple Relations in Graphs

- **Identity:**
  \[ g(x) = x \]

- **Translator:**
  \[ g(x | \Delta) = x + \Delta \]
  [Bordes et al. 13’]

- **Affine:**
  \[ g(x | A, \Delta) = Ax + \Delta \]
  [Nickel et al., 11’]

- **Diagonal:**
  \[ g(x | b) = b \odot x \]
  [Yang et al., 15’]

*Figure Credit: Alex Peysakhovich*
**TagSpace**

*Input*: restaurant has great food  
*Label*: #yum, #restaurant

*Use-cases:*
- Labeling posts
- Clustering of hashtags

Reference: [Weston et al. 14'], [Wu et al. 18']  
[https://github.com/facebookresearch/StarSpace](https://github.com/facebookresearch/StarSpace)

---

**PageSpace**

*Input*: (user, page) pairs

*Use-cases:*
- Clustering of pages
- Recommending pages to users

---

Application: TagSpace, PageSpace
Application: world2vec

Slide Credit: Alex Peysakhovich
The Power of Universal Behavioral Features

- What pages or topics might you be interested in?
- Which posts contain misinformation, hate speech, election interference, …?
- Is a person’s account fake / hijacked?
- What songs might you like? (even if you’ve never provided any song info)

---

Application: world2vec

*Slide Credit: Adam Lerer*
Reinforcement Learning Introduction
Reinforcement Learning

Evaluative feedback in the form of reward

No supervision on the right action

Types of Machine Learning

Supervised Learning
- Train Input: \( \{X, Y\} \)
- Learning output: \( f : X \to Y, P(y|x) \)
- E.g. classification

Unsupervised Learning
- Input: \( \{X\} \)
- Learning output: \( P(x) \)
- Example: Clustering, density estimation, etc.

Supervised Learning
- Train Input: \( \{X, Y\} \)
- Learning output: \( f : X \to Y, P(y|x) \)
- E.g. classification

Unsupervised Learning
- Input: \( \{X\} \)
- Learning output: \( P(x) \)
- Example: Clustering, density estimation, etc.

Reinforcement Learning
- Evaluative feedback in the form of reward
- No supervision on the right action
**RL:** Sequential decision making in an environment with evaluative feedback.

- Environment may be unknown, non-linear, stochastic and complex.
- Agent learns a policy to map states of the environments to actions.
  - Seeking to maximize cumulative reward in the long run.

---

**What is Reinforcement Learning?**

---

**Figure Credit:** Rich Sutton
What is Reinforcement Learning?

**RL:** Sequential decision making in an environment with evaluative feedback.

**Evaluative Feedback**
- Pick an action, receive a reward (positive or negative)
- No supervision for what the “correct” action is or would have been, unlike supervised learning

**Sequential Decisions**
- Plan and execute actions over a sequence of states
- Reward may be delayed, requiring optimization of future rewards (long-term planning).
**RL: Environment Interaction API**

At each time step $t$, the agent:
- Receives observation $o_t$
- Executes action $a_t$

At each time step $t$, the environment:
- Receives action $a_t$
- Emits observation $o_{t+1}$
- Emits scalar reward $r_{t+1}$

Slide credit: David Silver
Signature Challenges in Reinforcement Learning

- Evaluative feedback: Need trial and error to find the right action
- Delayed feedback: Actions may not lead to immediate reward
- Non-stationarity: Data distribution of visited states changes when the policy changes
- Fleeting nature of time and online data

Slide adapted from: Richard Sutton
Robot Locomotion

- **Objective**: Make the robot move forward
- **State**: Angle and position of the joints
- **Action**: Torques applied on joints
- **Reward**: +1 at each time step upright and moving forward

Figures copyright John Schulman et al., 2016. Reproduced with permission.

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n
Atari Games

- **Objective**: Complete the game with the highest score
- **State**: Raw pixel inputs of the game state
- **Action**: Game controls e.g. Left, Right, Up, Down
- **Reward**: Score increase/decrease at each time step

Figures copyright Volodymyr Mnih et al., 2013. Reproduced with permission.

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n
Examples of RL tasks

Go

- **Objective**: Defeat opponent
- **State**: Board pieces
- **Action**: Where to put next piece down
- **Reward**: +1 if win at the end of game, 0 otherwise

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n
Markov Decision Processes
Markov Decision Processes (MDPs)

- MDPs: Theoretical framework underlying RL
**MDPs**: Theoretical framework underlying RL

An MDP is defined as a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{R}, \mathbb{T}, \gamma)$

- $\mathcal{S}$: Set of possible states
- $\mathcal{A}$: Set of possible actions
- $\mathcal{R}(s, a, s')$: Distribution of reward
- $\mathbb{T}(s, a, s')$: Transition probability distribution, also written as $p(s'|s,a)$
- $\gamma$: Discount factor
**MDPs**: Theoretical framework underlying RL

An MDP is defined as a tuple \((S, A, R, T, \gamma)\)

- \(S\) : Set of possible states
- \(A\) : Set of possible actions
- \(R(s, a, s')\) : Distribution of reward
- \(T(s, a, s')\) : Transition probability distribution, also written as \(p(s'|s,a)\)
- \(\gamma\) : Discount factor

**Interaction trajectory**: \(\ldots s_t, a_t, r_{t+1}, s_{t+1}, a_{t+1}, r_{t+2}, s_{t+2}, \ldots\)
MDPs: Theoretical framework underlying RL

An MDP is defined as a tuple \((S, A, R, T, \gamma)\)

- \(S\) : Set of possible states
- \(A\) : Set of possible actions
- \(R(s, a, s')\) : Distribution of reward
- \(T(s, a, s')\) : Transition probability distribution, also written as \(p(s'|s, a)\)
- \(\gamma\) : Discount factor

Interaction trajectory: \(\ldots S_t, a_t, r_{t+1}, S_{t+1}, a_{t+1}, r_{t+2}, S_{t+2}, \ldots\)

Markov property: Current state completely characterizes state of the environment

Assumption: Most recent observation is a sufficient statistic of history
\[
p(S_{t+1} = s'|S_t = s_t, A_t = a_t, S_{t-1} = s_{t-1}, \ldots S_0 = s_0) = p(S_{t+1} = s'|S_t = s_t, A_t = a_t)
\]
### MDP Variations

**Fully observed MDP**
- Agent receives the true state $s_t$ at time $t$
- Example: Chess, Go

**Partially observed MDP**
- Agent perceives its own partial observation $o_t$ of the state $s_t$ at time $t$, using past states e.g. with an RNN
- Example: Poker, First-person games (e.g. Doom)
### MDP Variations

**Fully observed MDP**
- Agent receives the true state $s_t$ at time $t$
- Example: Chess, Go

**Partially observed MDP**
- Agent perceives its own partial observation $o_t$ of the state $s_t$ at time $t$, using past states.

We will assume **fully observed MDPs** for this lecture.
In Reinforcement Learning, we assume an underlying MDP with unknown:
- Transition probability distribution $T$
- Reward distribution $R$

MDPs in the context of RL
In Reinforcement Learning, we assume an underlying MDP with unknown:

- Transition probability distribution $\mathbb{T}$
- Reward distribution $\mathcal{R}$

Evaluative feedback comes into play, trial and error necessary
In Reinforcement Learning, we assume an underlying MDP with unknown:

- Transition probability distribution $T$
- Reward distribution $R$

Evaluative feedback comes into play, trial and error necessary.

For this and next lecture, assume that we know the true reward and transition distribution and look at algorithms for solving MDPs i.e. finding the best policy:

- Rewards known everywhere, no evaluative feedback
- Know how the world works i.e. all transitions

MDPs in the context of RL
A Grid World MDP

- Agent lives in a 2D grid environment
- State: Agent's 2D coordinates
- Actions: N, E, S, W
- Rewards: +1/-1 at absorbing states
- Walls block agent's path
- Actions to not always go as planned
  - 20% chance that agent drifts one cell left or right of direction of motion (except when blocked by wall).

Figure credits: Pieter Abbeel
Agent lives in a 2D grid environment
Agent lives in a 2D grid environment

State: Agent’s 2D coordinates

Actions: N, E, S, W

Rewards: +1/-1 at absorbing states

Figure credits: Pieter Abbeel
Agent lives in a 2D grid environment

State: Agent’s 2D coordinates
Actions: N, E, S, W
Rewards: +1/-1 at absorbing states

Walls block agent’s path
Actions to not always go as planned
- 20% chance that agent drifts one cell left or right of direction of motion (except when blocked by wall).

Figure credits: Pieter Abbeel