CIS 6930 Topics in Computing for Data Science
Week 11: More Deep Learning Topics (1)

Tue 11/16/2021

Yoshihiko (Yoshi) Suhara

2pm-3:20pm & 3:30pm-4:50pm

Course Schedule

- (Today) Tue 11/16 **(2x 80min)**
- Thu 11/18 **(2x 80min)** (* Finalize the term project title & plan!)
- Tue 11/23 Project presentation (1)
- Thu 11/30 Project presentation (2) (* Final session! 🥲)

Term Project
Project Presentation & Project Report

Term Project

- ~~Fri 11/12: Feedback & Pre-finalize the project plan~~
- **Fri 11/18:** Finalize the project title and plan
- Tue 11/23: Project presentations (1)
- Tue 11/30: Project presentations (2)
- **Tue 12/7:** Project report deadline

Each student must work on their own term project (**No group work**)

Did you receive an email from me? Please reply to the e-mail

Project Presentation (5min + 3min QA)
Tue 11/23 & Tue 11/30

The presentation should contain

- 1) Motivation (30 seconds)
- 2) Clear problem definition (1 minute)
- 3) Methodology (1-2 minutes)
- 4) Evaluation: Results and Discussion (1-2 minutes)
- 5) Key takeaway (30 seconds)

Notes

- Presentations will NOT be evaluated by the performance of your solution
- Students are encouraged to share in-progress work to get feedback from the instructor
- Please use this opportunity to improve the quality of your final project report

Project Report (due Tue 12/7)

The project report must contain

- 1) Motivation
- 2) Clear problem definition + related background
- 3) Methodology: Description of the solution/baselines and the dataset
- 4) Evaluation: Results and Discussion
- 5) Personal reflection: Challenges encountered, lessons learned

The ideal project report should also answer the following questions

- How is your solution different from other solutions (baselines)?
- How better is your solution than the baselines?
- Why is your solution better than the baselines?

Any Questions?

Today’s Agenda

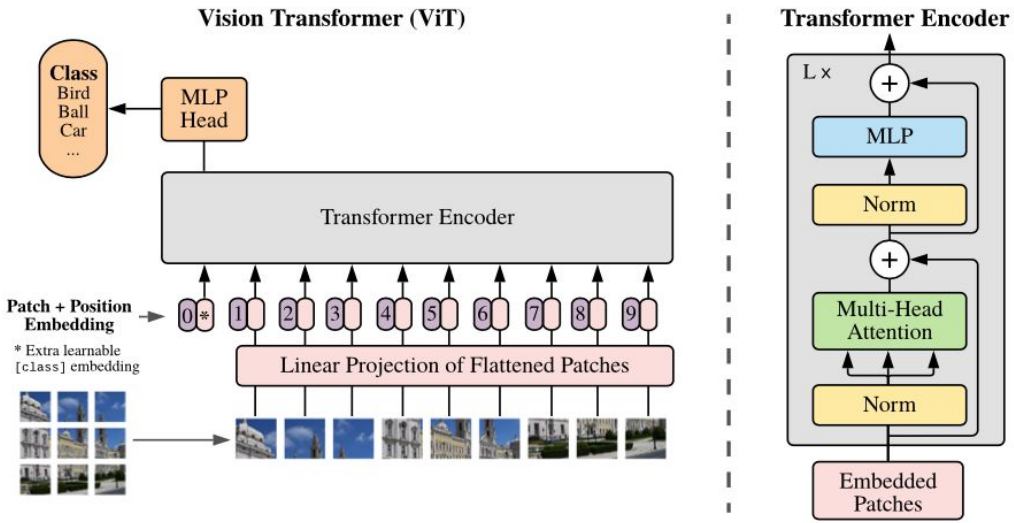
- Part I
  - Transformers for Computer Vision
  - Transformer Decoder-only Models (e.g., GPT-2/3)
  - Transformer Encoder-Decoder Models (e.g., BART, T5)

- Part II
  - Sparse Attention
  - Transfer Learning & Fine-tuning
  - Advanced Optimization Techniques

Transformers for Computer Vision

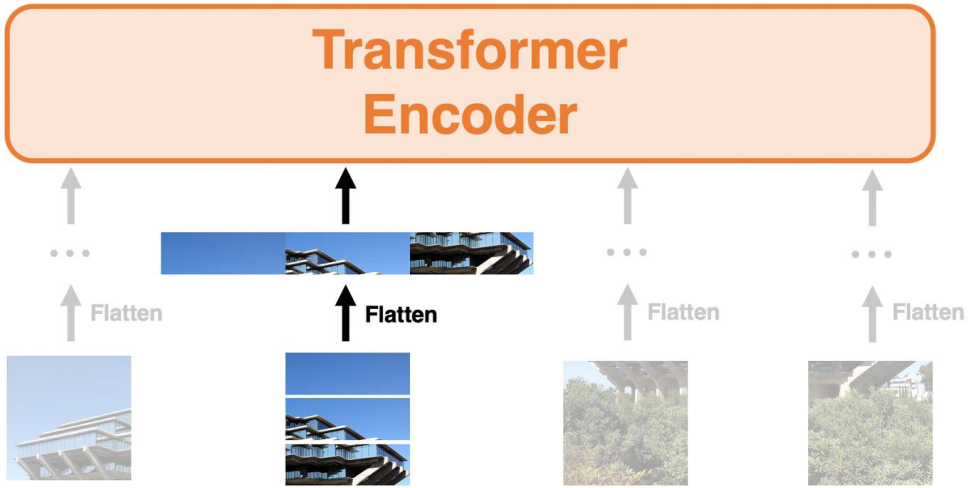
Vision Transformer (ViT) [Dosovitskiy et al. 2021]

- RGB Image → N Patches → Flattening patches
  - Each patch = (3 * P^2)-dimensional vector



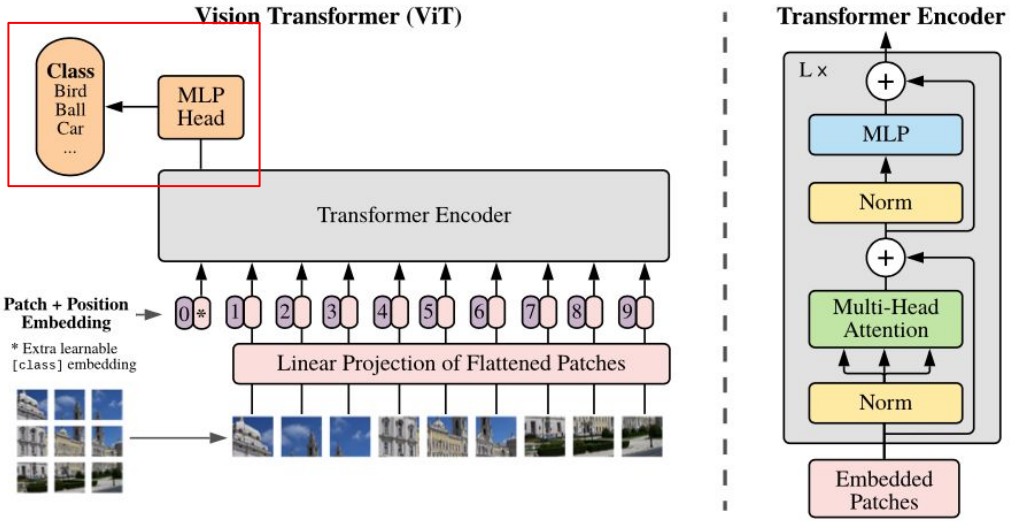
A. Dosovitskiy et al. “An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale”, ICLR ‘21

Input to Vision Transformer



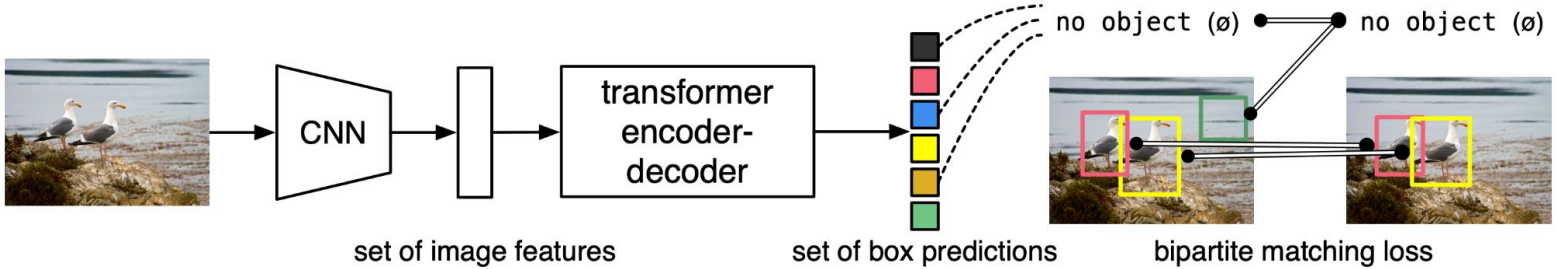
Vision Transformer Pre-training

- **“Supervised”** pre-training
- on the JFT-300M private image dataset
  - 18k image categories (!)



Detection Transformer (DETR) [Carion et al. 2020]

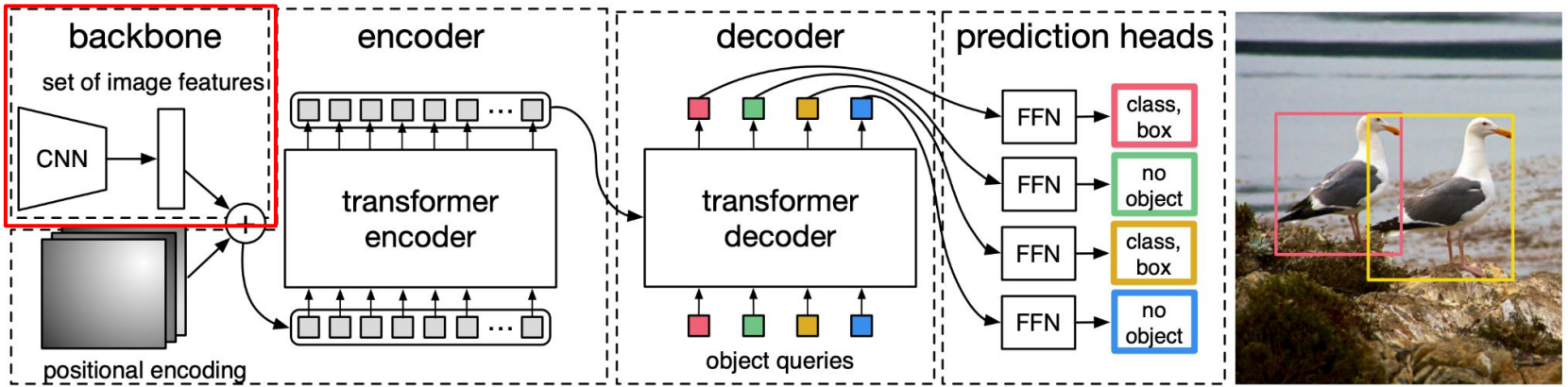
- CNN + Transformer for object detection



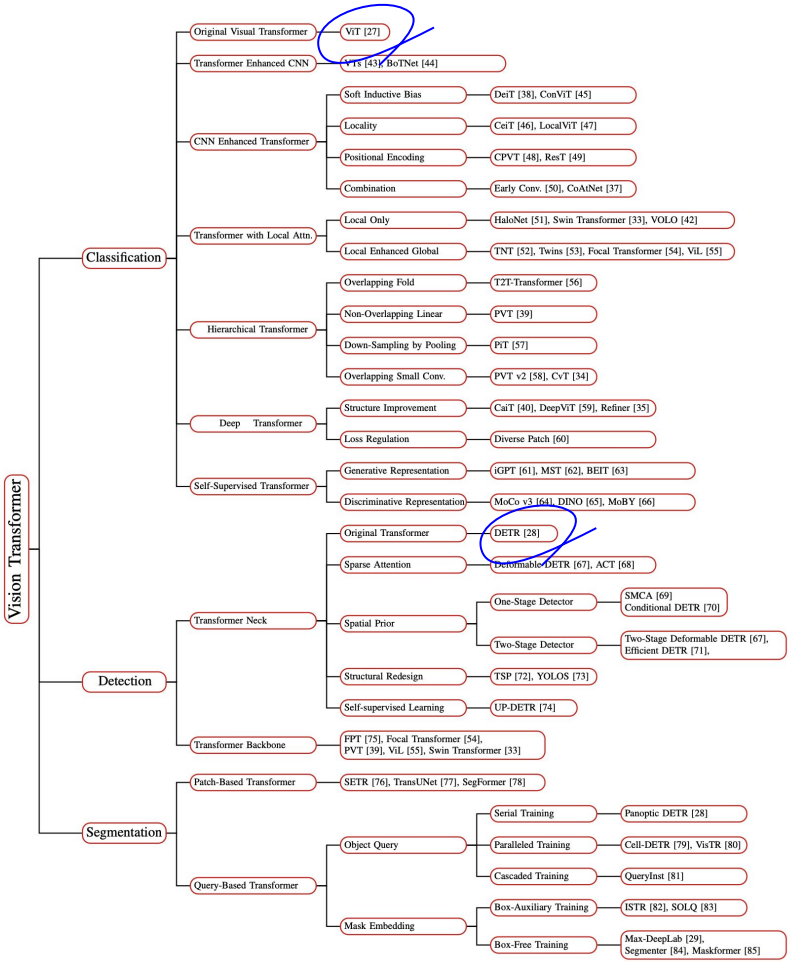
N. Carion et al. “End-to-End Object Detection with Transformers”, arXiv 2020.

Detection Transformer (DETR) [Carion et al. 2020]

- CNN + Transformer for object detection



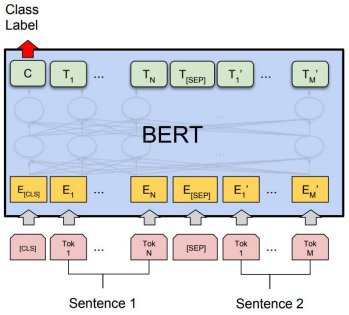
N. Carion et al. “End-to-End Object Detection with Transformers”, arXiv 2020.



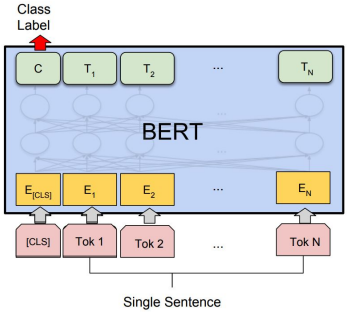
[2111.06091] A Survey of Visual Transformers

Going back to NLP...

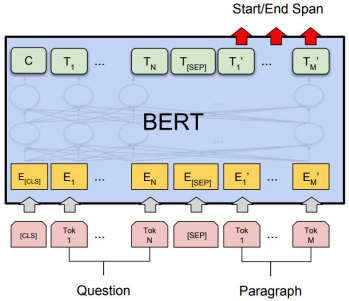
What are the limitations of BERT?



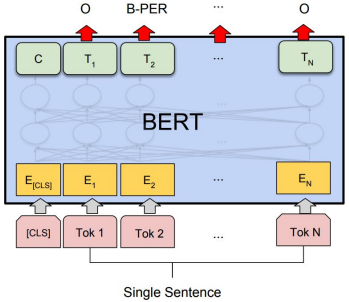
(a) Sentence Pair Classification Tasks: MNLI, QQP, QNLI, STS-B, MRPC, RTE, SWAG



(b) Single Sentence Classification Tasks: SST-2, CoLA



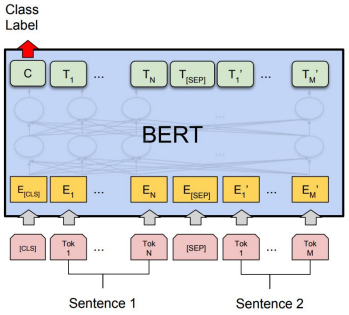
(c) Question Answering Tasks: SQuAD v1.1



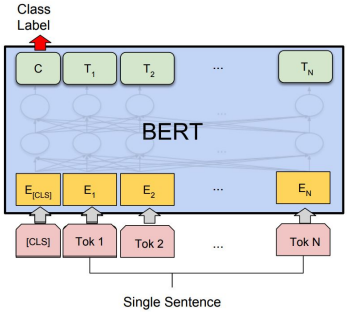
(d) Single Sentence Tagging Tasks: CoNLL-2003 NER

It looks versatile. Really? 🤔

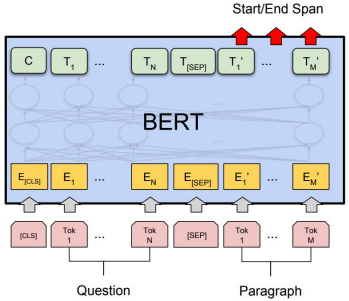
What are the limitations of BERT?



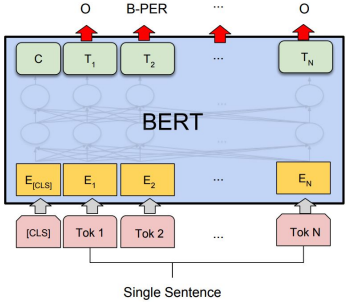
(a) Sentence Pair Classification Tasks: MNLI, QQP, QNLI, STS-B, MRPC, RTE, SWAG



(b) Single Sentence Classification Tasks: SST-2, CoLA



(c) Question Answering Tasks: SQuAD v1.1



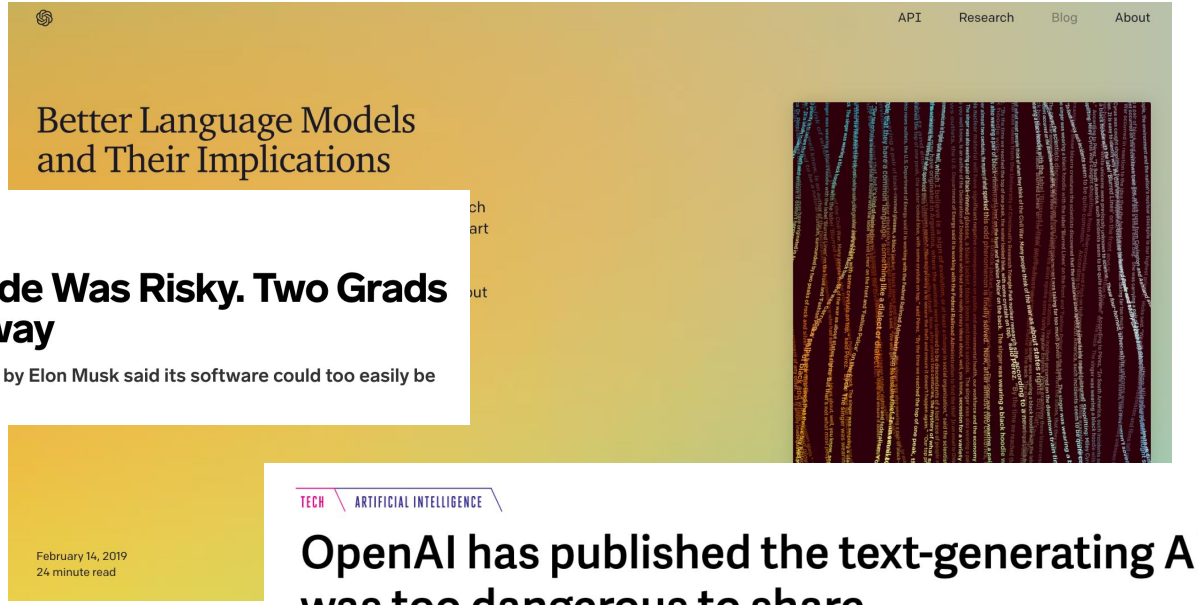
(d) Single Sentence Tagging Tasks: CoNLL-2003 NER

It looks versatile. Really? 🤔

Can it be used for text generation?

GPT-2

https://openai.com/blog/better-language-models/

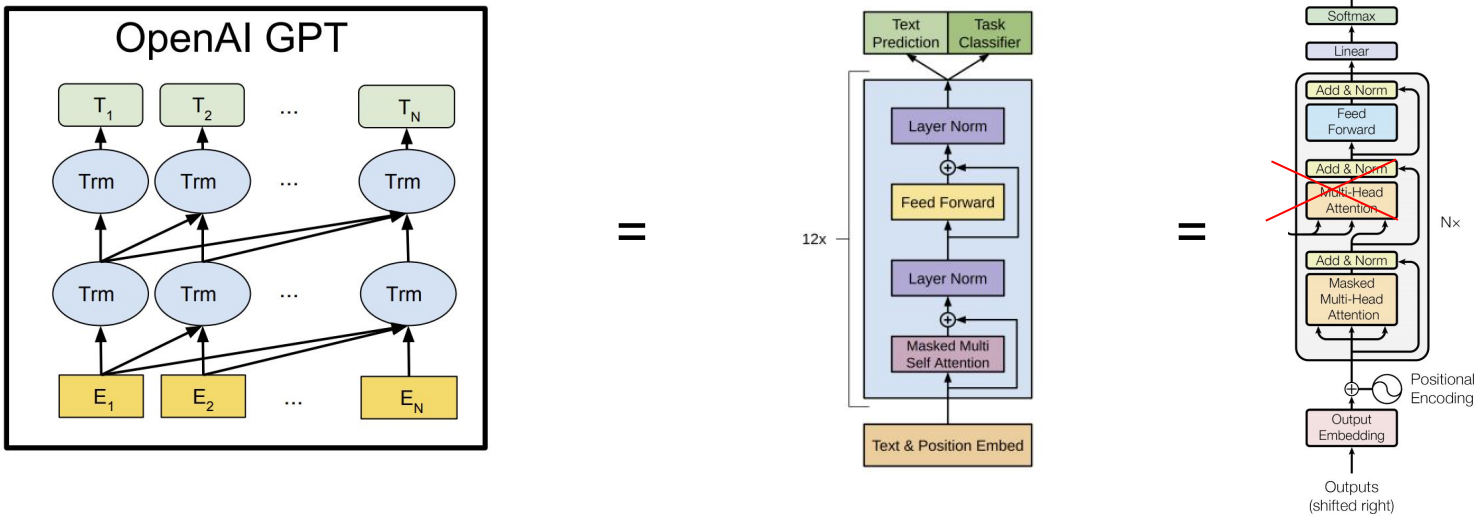


GPT History

- 2018: GPT (117M parameters)
- 2019: GPT-2 (1.5B parameters)
- 2020: GPT-3 (175B parameters)
- Coming soon?: GPT-4
  - https://towardsdatascience.com/gpt-4-will-have-100-trillion-parameters-500x-the-size-of-gpt-3-582b98d82253
  -

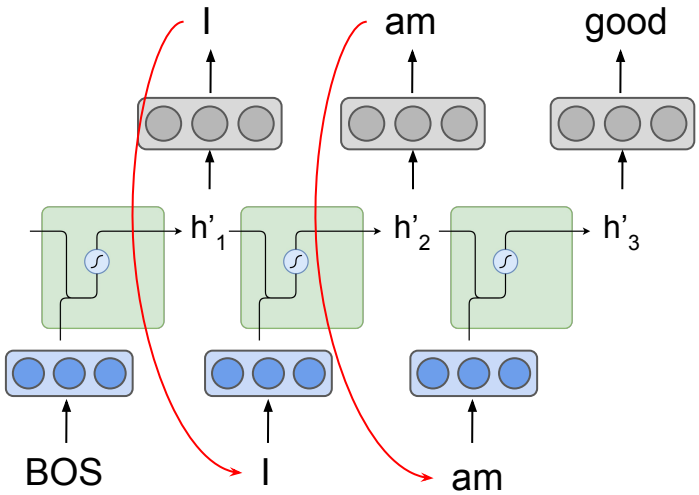
OpenAI GPT (Generative Pre-trained Transformer)

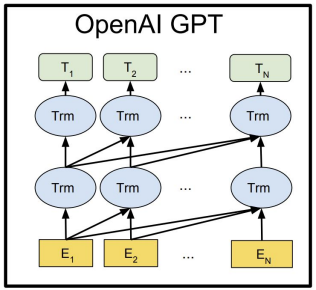
- GPT = Stacked **Transformer Decoder-only** Model
  - Autoregressive model

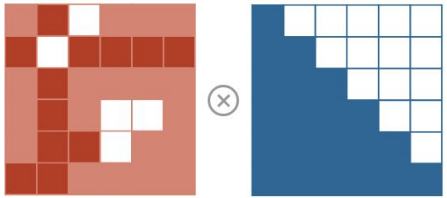


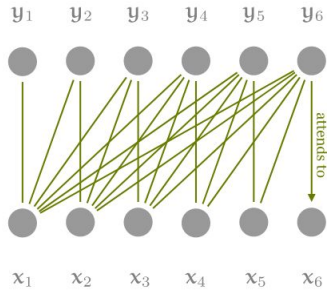
Technically, it does not use Encoder-Decoder Attention

Autoregressive Model







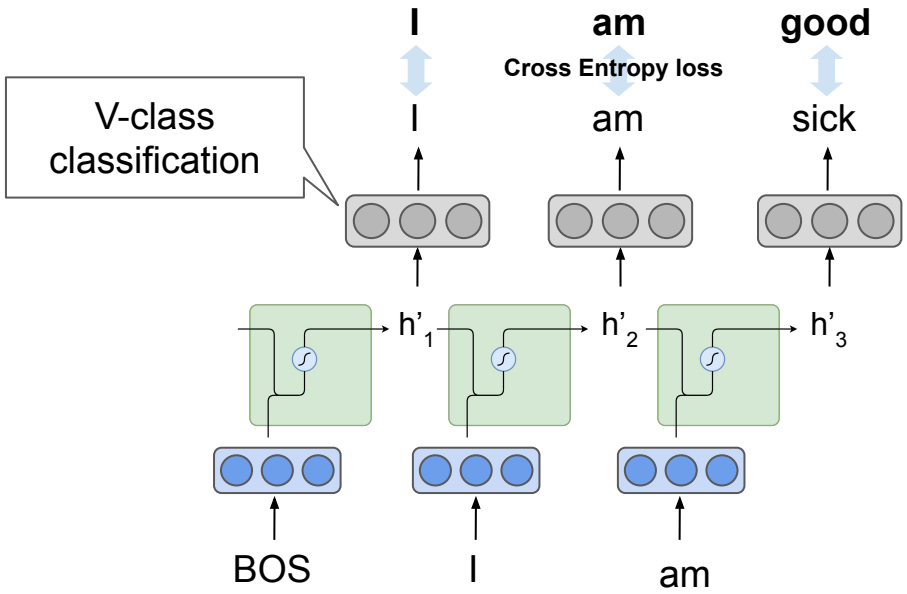


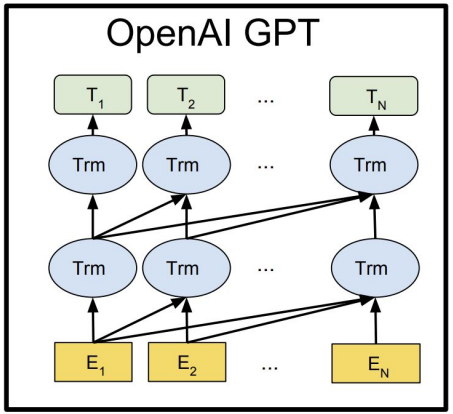
Decoder

Pre-training Corpus: BooksCorpus

- **7,000 books** in various genres

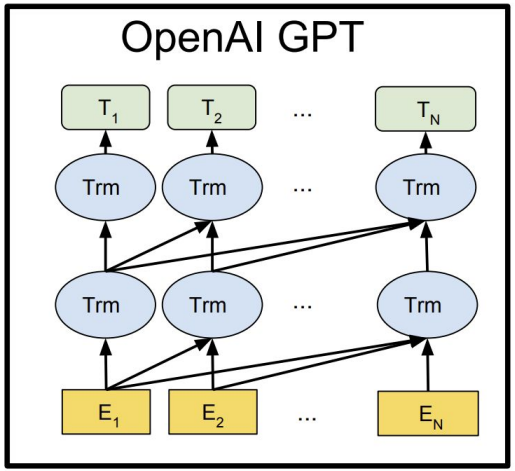
Pre-training: Causal Language Model = Teacher forcing





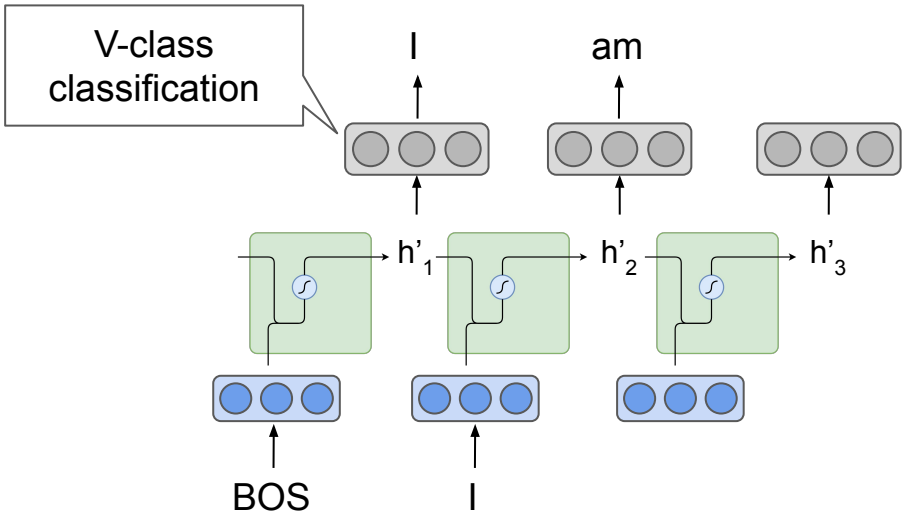
Can we Fine-tune GPT-2?

- Yes, do the same thing as Causal Language Model
  - 1) Give **input sequence** as prefix (i.e., not used for loss calculation)
  - 2) Train the model with teacher forcing using the **target sequence**



Autoregressive LM as a Next Token Predictor

- It returns the “most likely” token given the context
  - e.g., $P(X_t \mid X_{t<}$= “<BOS> I”)

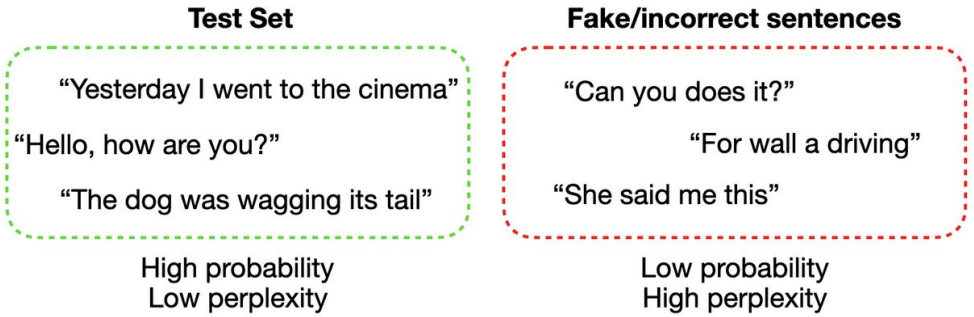


How do we evaluate Autoregressive LMs? (1/2)

- Hold out sentences as a test set (don’t use them for fine-tuning!)
- Fine-tune an LM using the training set
- Evaluate if the fine-tuned LM returns **high probabilities** for the test set

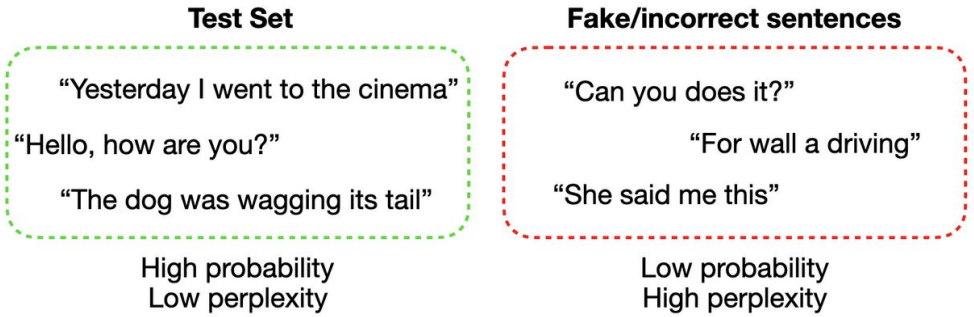
How do we evaluate Autoregressive LMs? (2/2)

- **Perplexity** is a commonly used metric



How do we evaluate Autoregressive LMs? (2/2)

- **Perplexity** is a commonly used metric

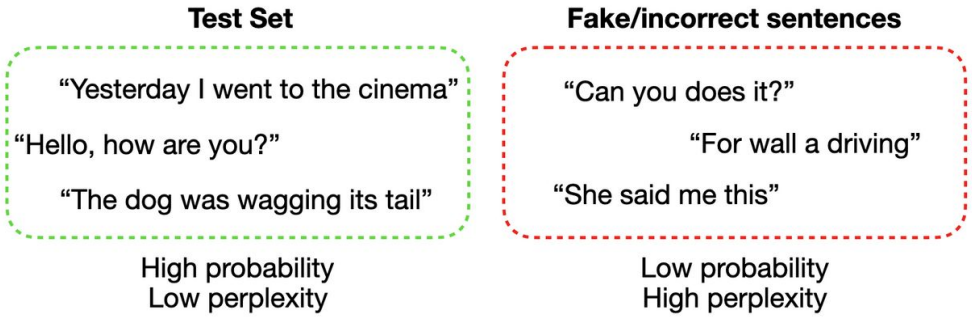


$$PP(W) = \sqrt[N]{\frac{1}{P(w_1, w_2, \ldots, w_N)}}$$ (n-th root of the joint probability)

$$P(W) = P(w_1, w_2, \ldots, w_N) = P(w_1)P(w_2)\ldots P(w_N) = \prod_{i=1}^{N} P(w_i)$$

How do we evaluate Autoregressive LMs? (2/2)

- **Perplexity** is a commonly used metric

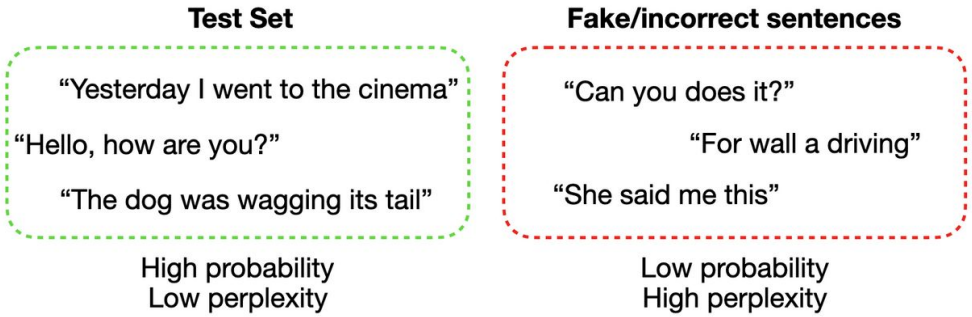


$$PP(W) = \sqrt[N]{\frac{1}{P(w_1, w_2, \ldots, w_N)}}$$ (n-th root of the joint probability)

$$P(W) = P(w_1, w_2, \ldots, w_N) = P(w_1)P(w_2)\ldots P(w_N) = \prod_{i=1}^{N} P(w_i)$$

How do we evaluate Autoregressive LMs? (2/2)

- **Perplexity** is a commonly used metric

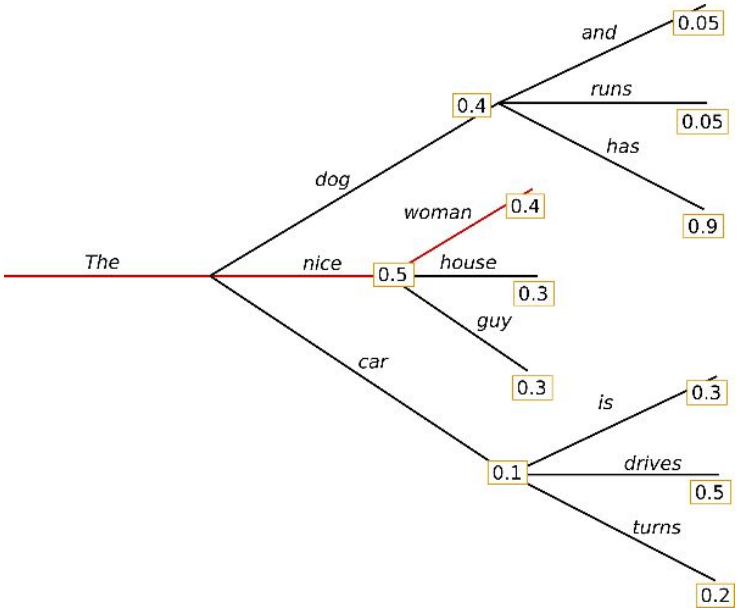


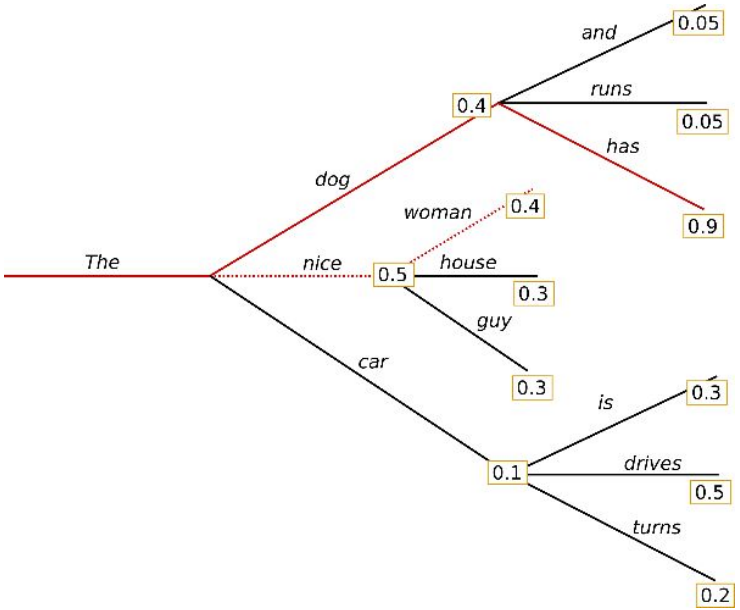
Perplexity can also be calculated as the exponential of the cross-entropy (easy!)

$$P(W) = P(w_1, w_2, \ldots, w_N) = P(w_1)P(w_2)\ldots P(w_N) = \prod_{i=1}^{N} P(w_i)$$

A little bit more about Decoding

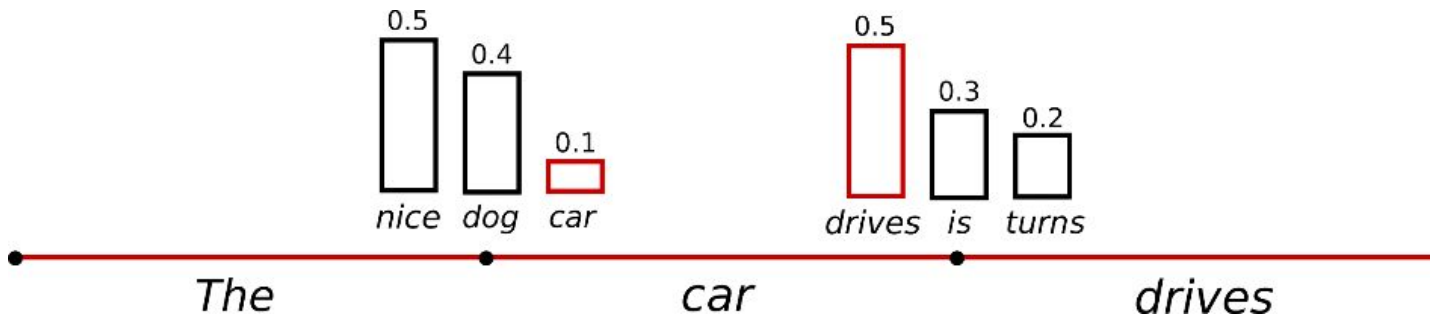
Decoding Algorithm: Greedy Search & Beam Search





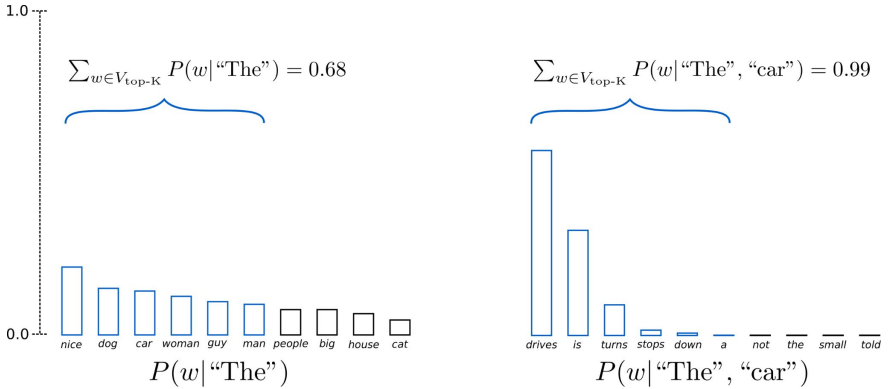
Yet Another Decoding Algorithm: Sampling

- At each step, **randomly choose a token** according to the token probability distribution
  - Note: we need to fix the random seed for reproducibility

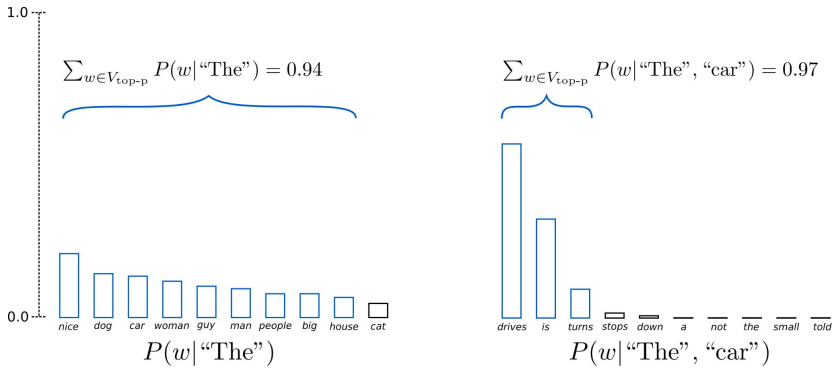


Top-k & Top-p (Nucleus) Sampling

- Use the top-k (the cumulative probability) to determine the token candidates



Top-k sampling



Top-p sampling

N-gram Blocking

- The technique excludes any sequence that contains the same N-gram tokens from the candidate sequence

- e.g., 2-gram blocking
  - The dog and the [ ]
    - ~~dog (0.55)~~
    - cat (0.30)
    - boy (0.10)
    - ...

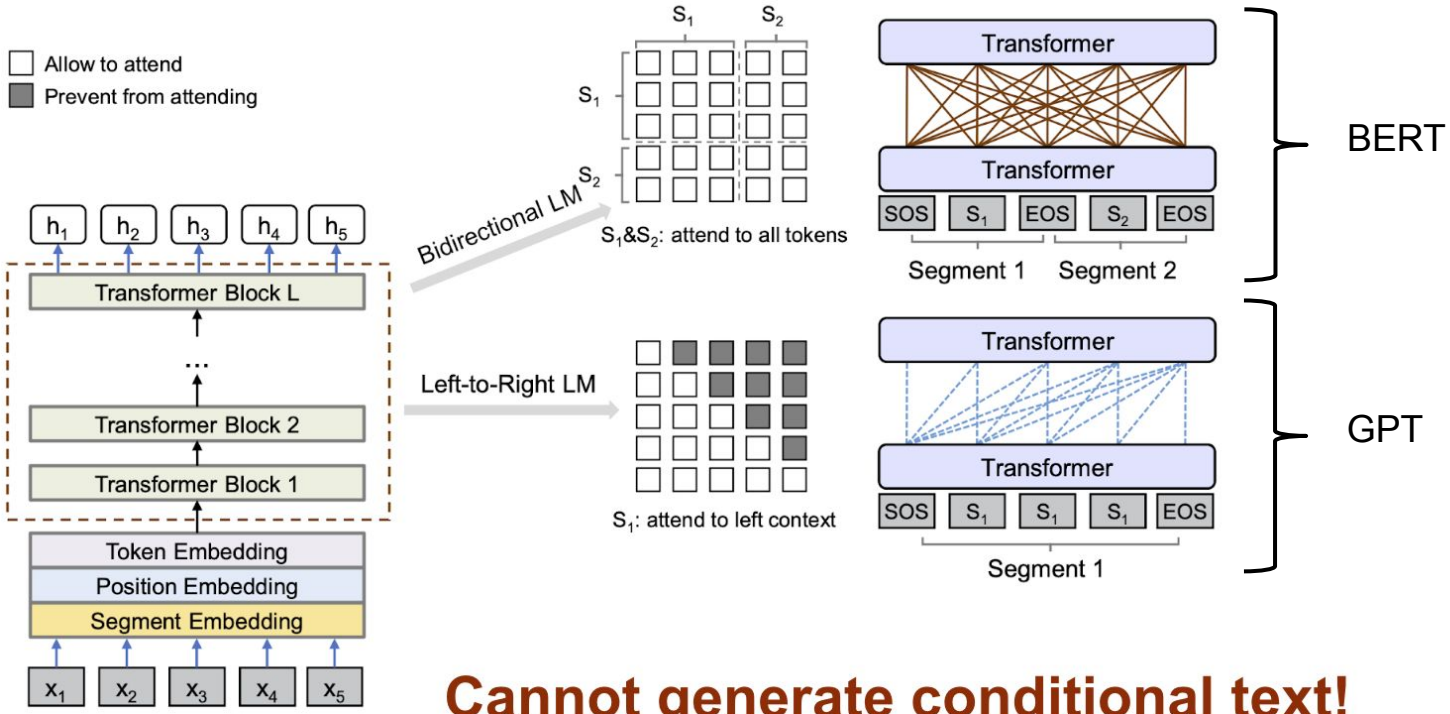
2 or 3-gram blocking are commonly used to avoid the repetition issue

Hands-on: Playing with GPT-2 & Decoding algorithms

- Google Colab

Pre-trained Encoder-Decoder Models

UniLM: Limitations of BERT & GPT



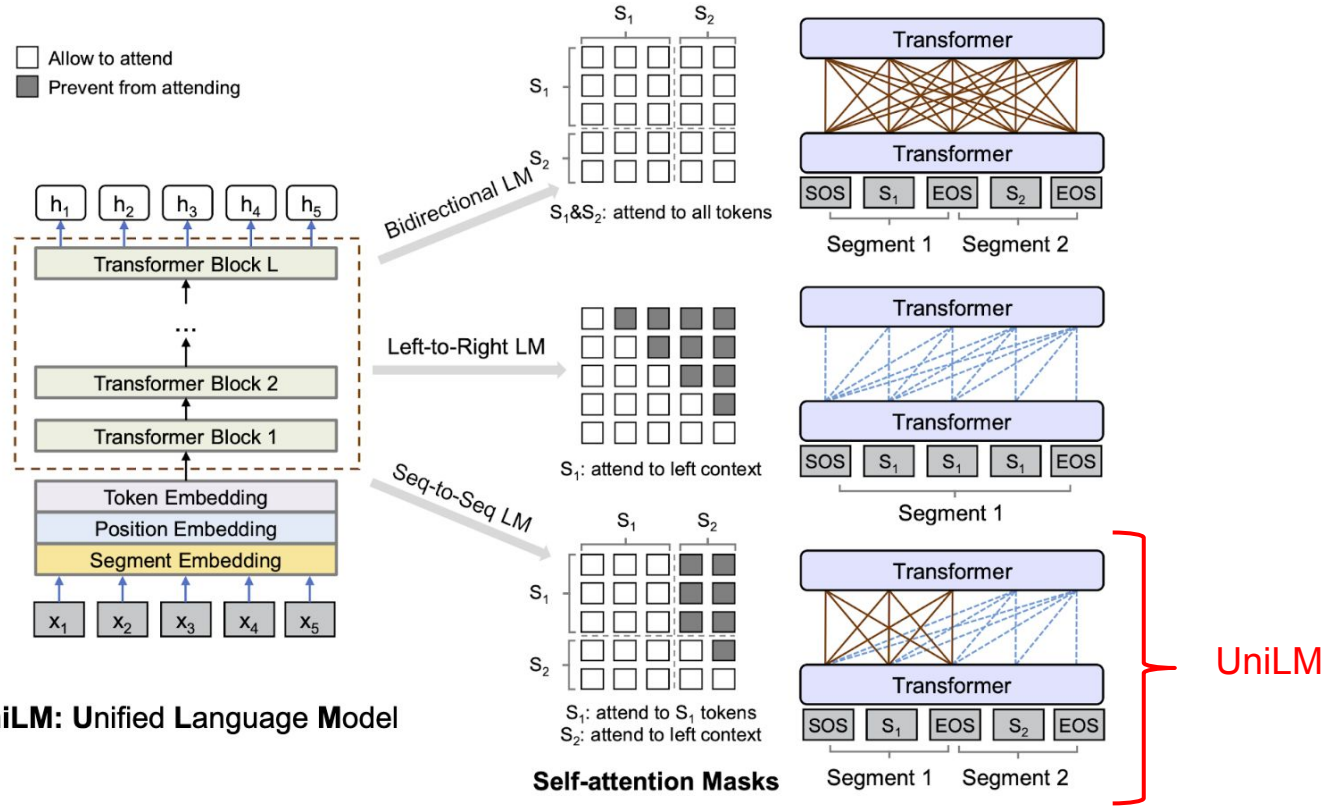
Cannot generate conditional text!

GPT: Generative Pre-Training

https://docs.google.com/presentation/d/1FWqQDqi1edA3ivobQk93gq_NBEITKq8WzG1RIJzEDJM/edit#slide=id.g59b5860c59_0_564

UniLM: Unified Language Model

UniLM = BERT + GPT

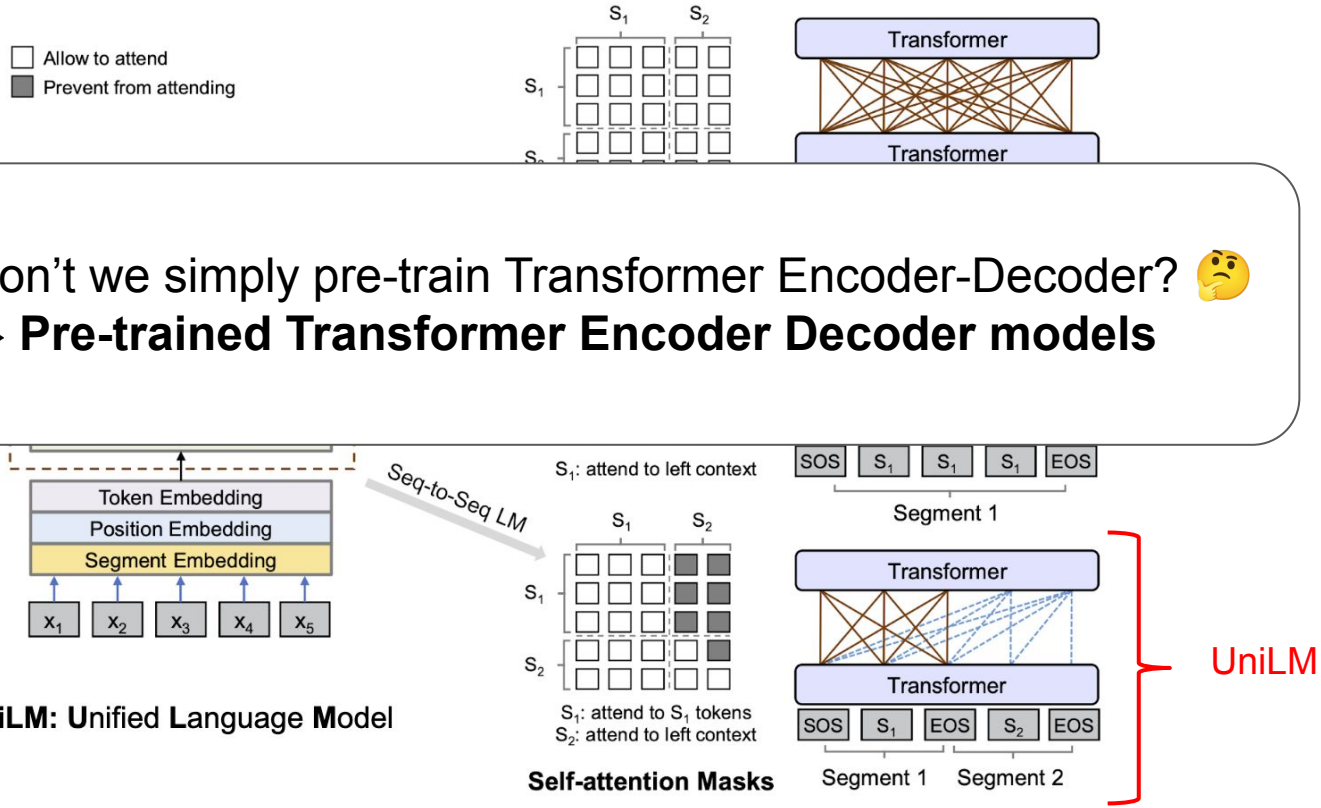


UniLM: Unified Language Model

https://docs.google.com/presentation/d/1FWqQDqi1edA3ivobQk93gq_NBEITKq8WzG1RIJzEDJM/edit#slide=id.g59b5860c59_0_564

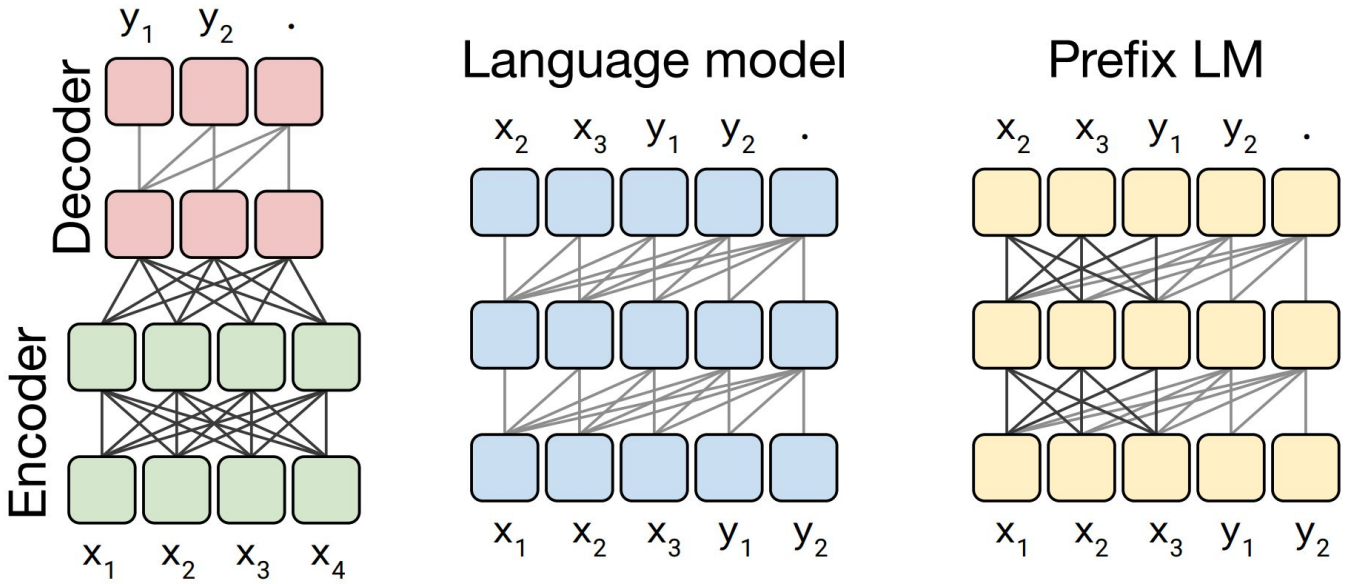
UniLM: Unified Language Model

UniLM = BERT + GPT



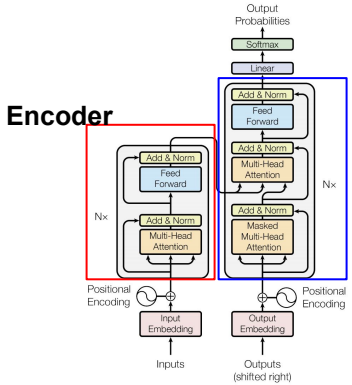
https://docs.google.com/presentation/d/1FWqQDqi1edA3ivobQk93gq_NBEITKq8WzG1RIJzEDJM/edit#slide=id.g59b5860c59_0_564

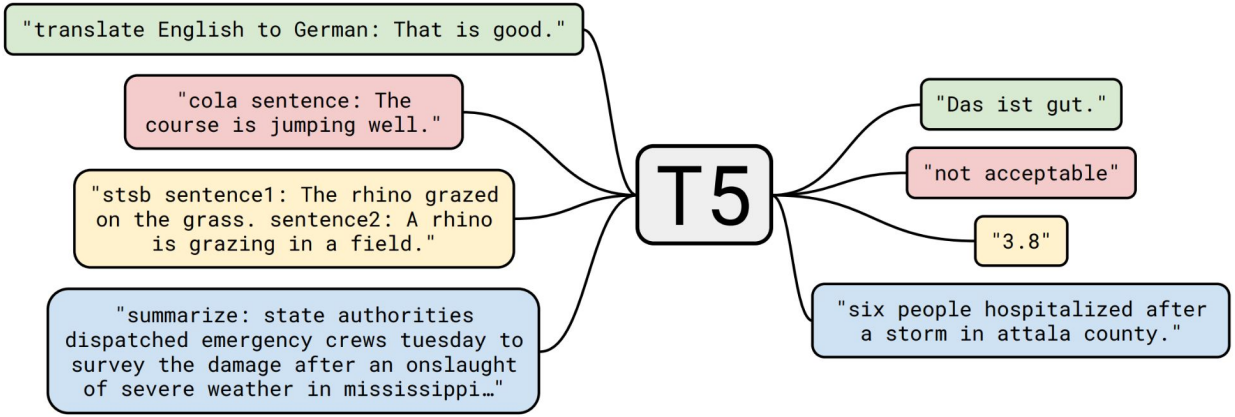
Encoder-Decoder vs Prefix LM



Text-to-Text Transfer Transformer (T5)

- Transformer Encoder-Decoder





T5's "Multi-task" Pre-training

- The Colossal Clean Crawled Corpus (C4): **20TB** web corpus
- Multiple pre-training methods

Objective	Inputs	Targets
Prefix language modeling	Thank you for inviting	me to your party last week .
BERT-style Devlin et al. (2018)	Thank you <M> <M> me to your party apple week .	*(original text)*
Deshuffling	party me for your to . last fun you inviting week Thank	*(original text)*
MASS-style Song et al. (2019)	Thank you <M> <M> me to your party <M> week .	*(original text)*
I.i.d. noise, replace spans	Thank you <X> me to your party <Y> week .	<X> for inviting <Y> last <Z>
I.i.d. noise, drop tokens	Thank you me to your party week .	for inviting last
Random spans	Thank you <X> to <Y> week .	<X> for inviting me <Y> your party last <Z>

More is Better … ? No

- We should let the model “read good books”

Data set	Size	GLUE	CNNDM	SQuAD	SGLUE	EnDe	EnFr	EnRo
★ C4	745GB	83.28	**19.24**	80.88	71.36	**26.98**	**39.82**	**27.65**
C4, unfiltered	6.1TB	81.46	19.14	78.78	68.04	26.55	39.34	27.21
RealNews-like	35GB	**83.83**	**19.23**	80.39	72.38	**26.75**	**39.90**	**27.48**
WebText-like	17GB	**84.03**	**19.31**	**81.42**	71.40	**26.80**	**39.74**	**27.59**
Wikipedia	16GB	81.85	**19.31**	81.29	68.01	**26.94**	39.69	**27.67**
Wikipedia + TBC	20GB	83.65	**19.28**	**82.08**	**73.24**	**26.77**	39.63	**27.57**

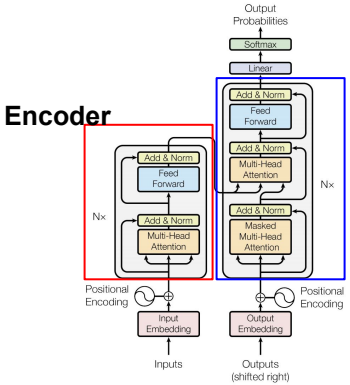
T5 Models

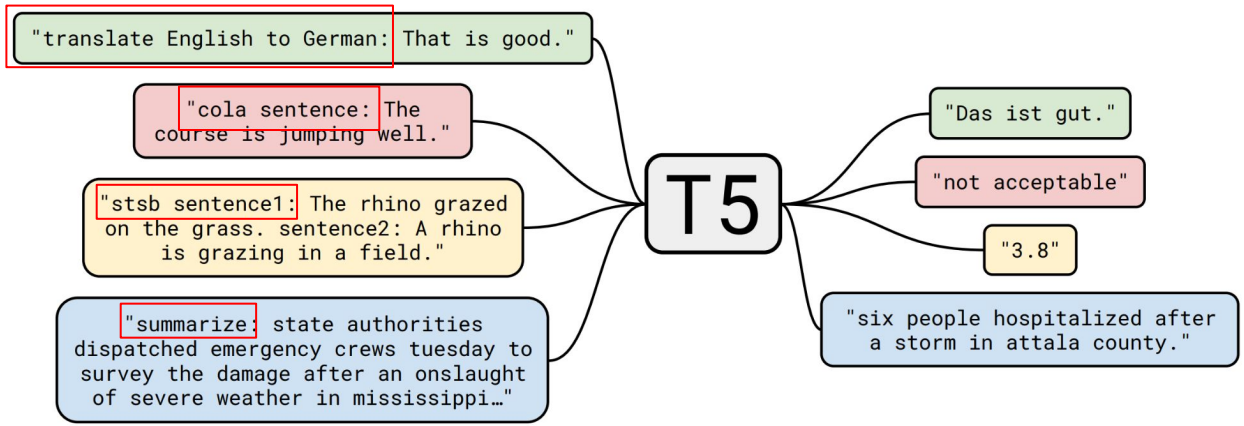
- Small (60M): d=512, L=6, H=8
- Base (220M): d=1024, L=12, H=16
- Large (770M): d=1024, L=24, H=16

Can you explain how the model designs look like?

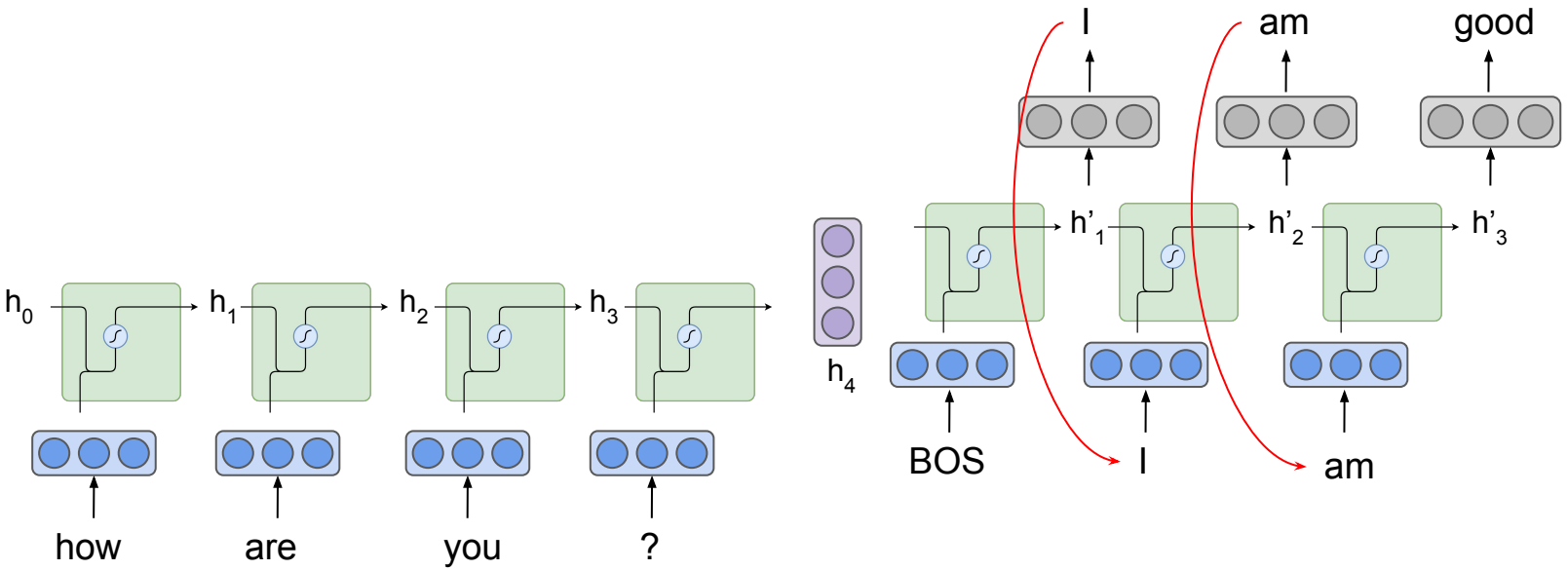
Text-to-Text Transfer Transformer (T5)

- Transformer Encoder-Decoder

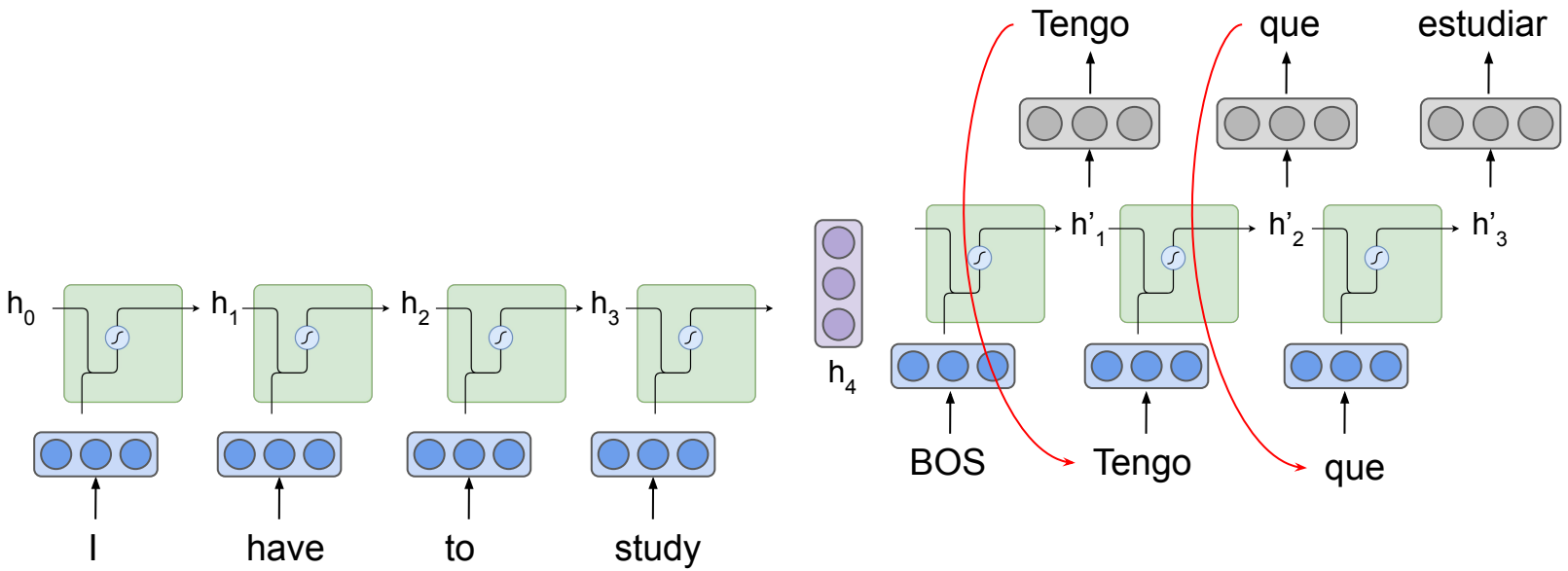




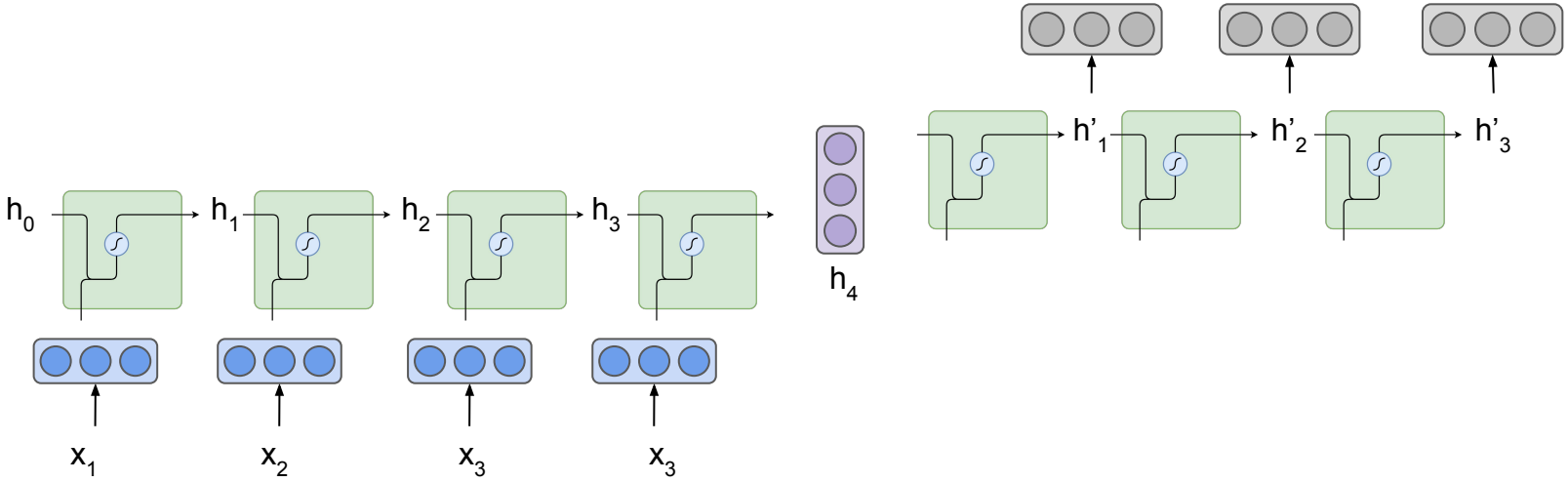
Example 1: Response Generation

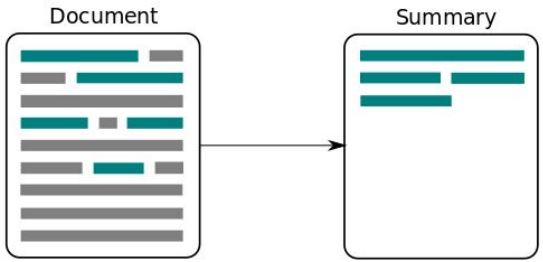


Example 2: Machine Translation



Example 3: Text Summarization





A List of Pre-trained Encoder-Decoder Models

- T5 by Google
- BART by Facebook
- MASS by Microsoft
- PEGASUS by Google
- MARGE by Facebook
- OPTIMUS by Microsoft
- ...

They are *all* Transformer Encoder-Decoder Models and can be used in the same manner

Hands-on: Playing with T5

- Google Colab

Hugging Face Model Hub

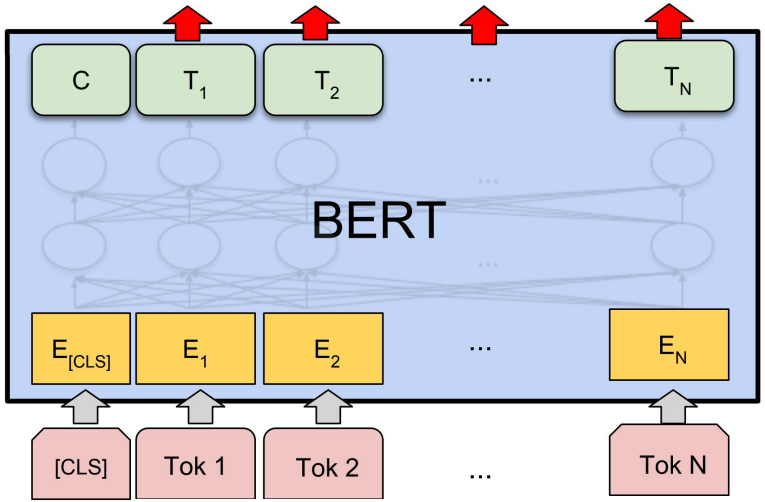
- https://huggingface.co/models
- Some models are **just pre-trained**. Some are **also fine-tuned**

Break?

Sparse Attention Models

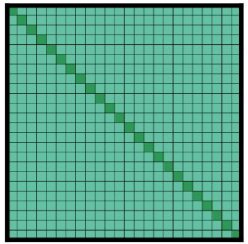
Limitations of Transformer Models

- Transformers **cannot take long sequences** as input due to **the quadratic dependency O(N^2)** on the sequence length (Memory capacity issue)

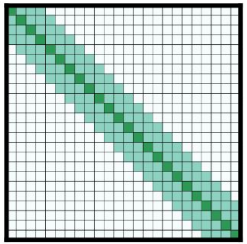


Longformer

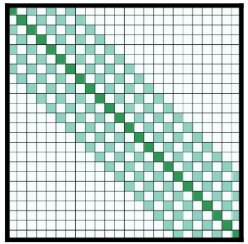
- Combination of multiple sparse attention mechanism



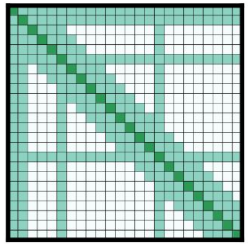
(a) Full n^2 attention



(b) Sliding window attention



(c) Dilated sliding window



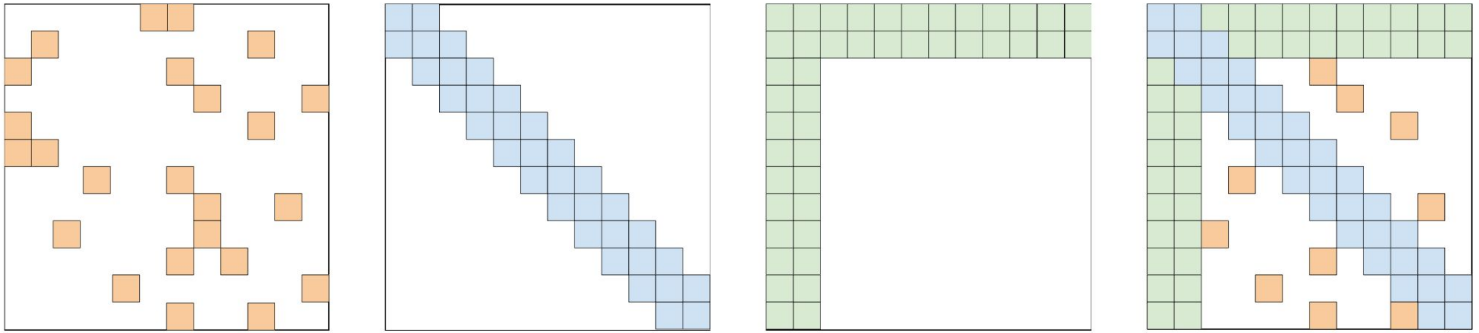
(d) Global+sliding window

Each layer does NOT have direct connections between all tokens anymore
But, high-order interactions can be computed **through multiple Transformer blocks**

[2004.05150] Longformer: The Long-Document Transformer

Big Bird

- Random attention + Two sparse attentions

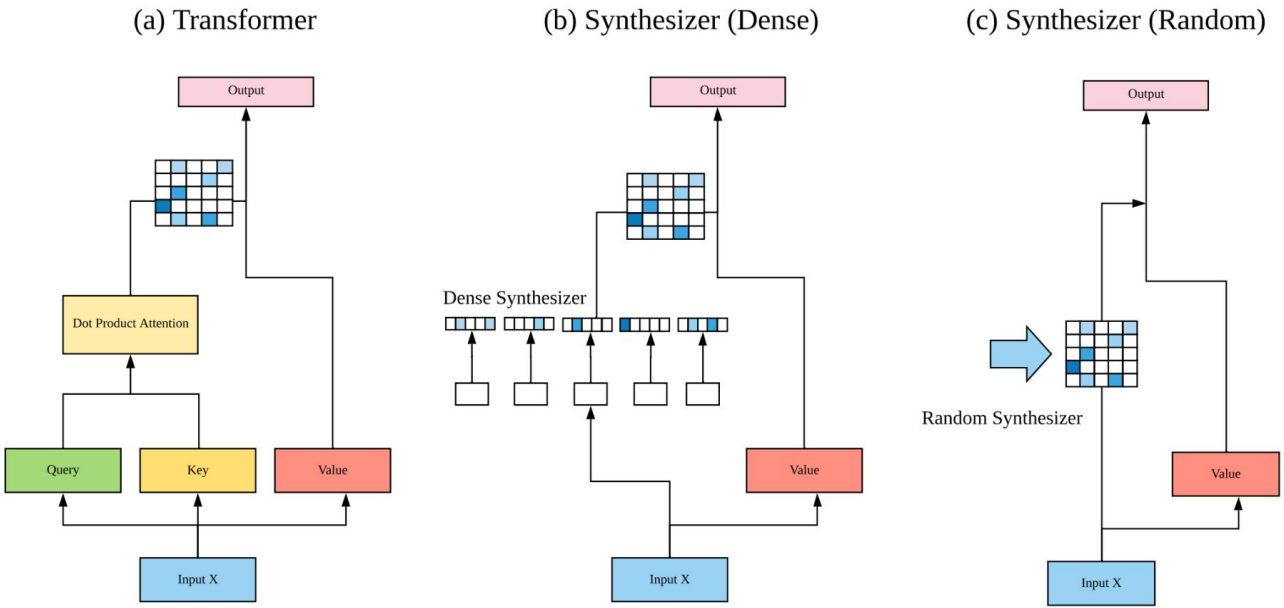


(a) Random attention (b) Window attention (c) Global Attention (d) BIGBIRD

[2007.14062] Big Bird: Transformers for Longer Sequences

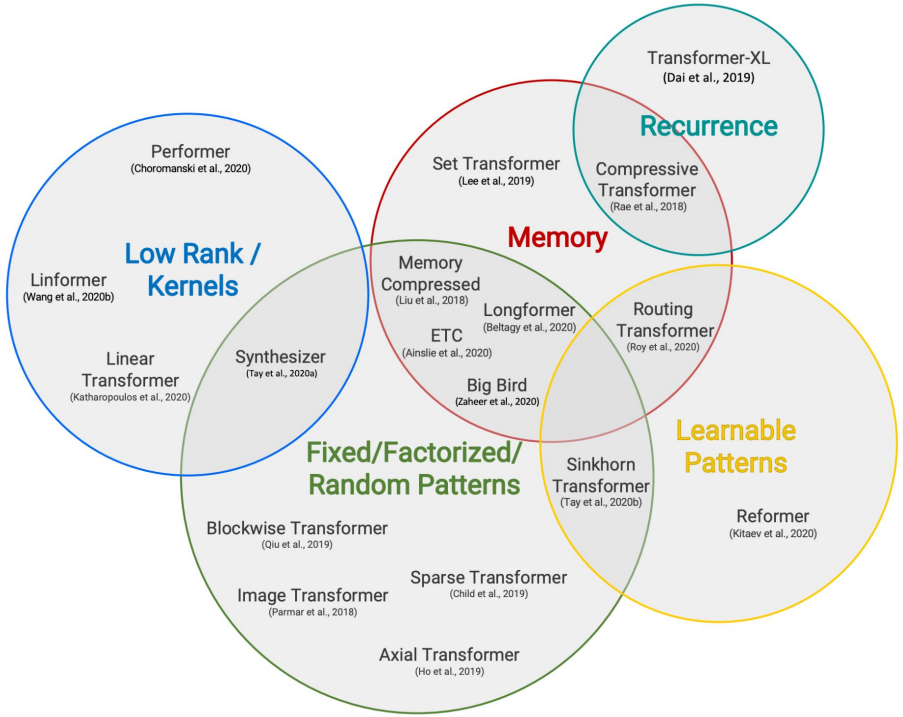
Synthesizer

- Outperforms T5 **without** “trainable” self-attention mechanism



[2005.00743] Synthesizer: Rethinking Self-Attention in Transformer Models

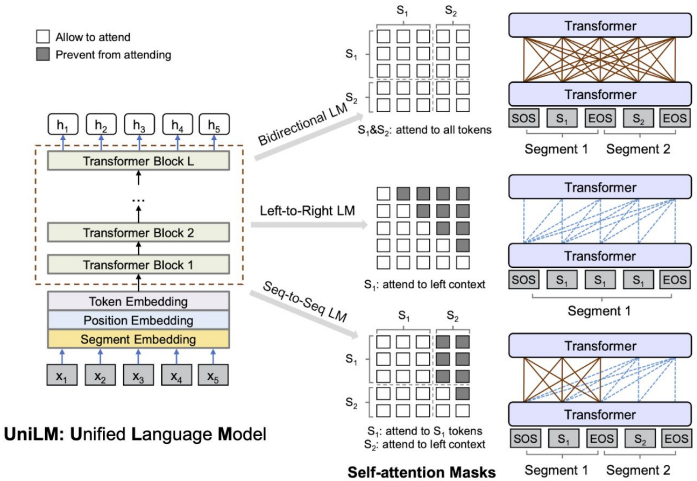
Efficient Transformers



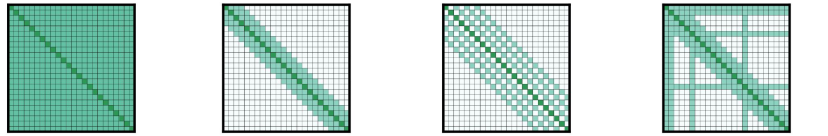
[2009.06732] Efficient Transformers: A Survey

Summary: Transformers!

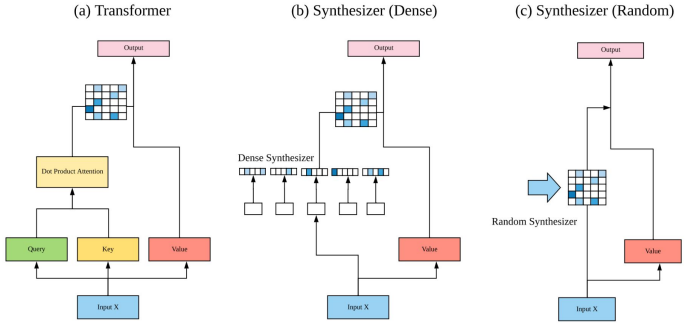
- The attention mechanism **design** is everything



UniLM: Unified Language Model



(a) Full n^2 attention (b) Sliding window attention (c) Dilated sliding window (d) Global+sliding window

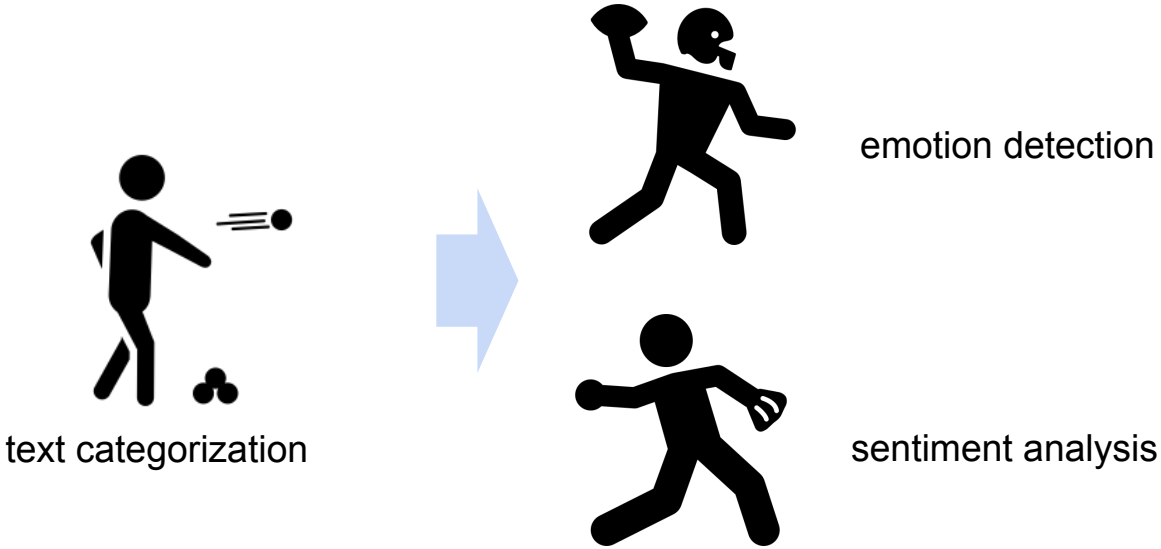


Any Questions?

Transfer Learning & Multi-task Learning

Transfer Learning

- A machine learning framework that trains a model on relevant task(s) before training the model on the target task



Is fine-tuning a pre-trained LM considered transfer learning?

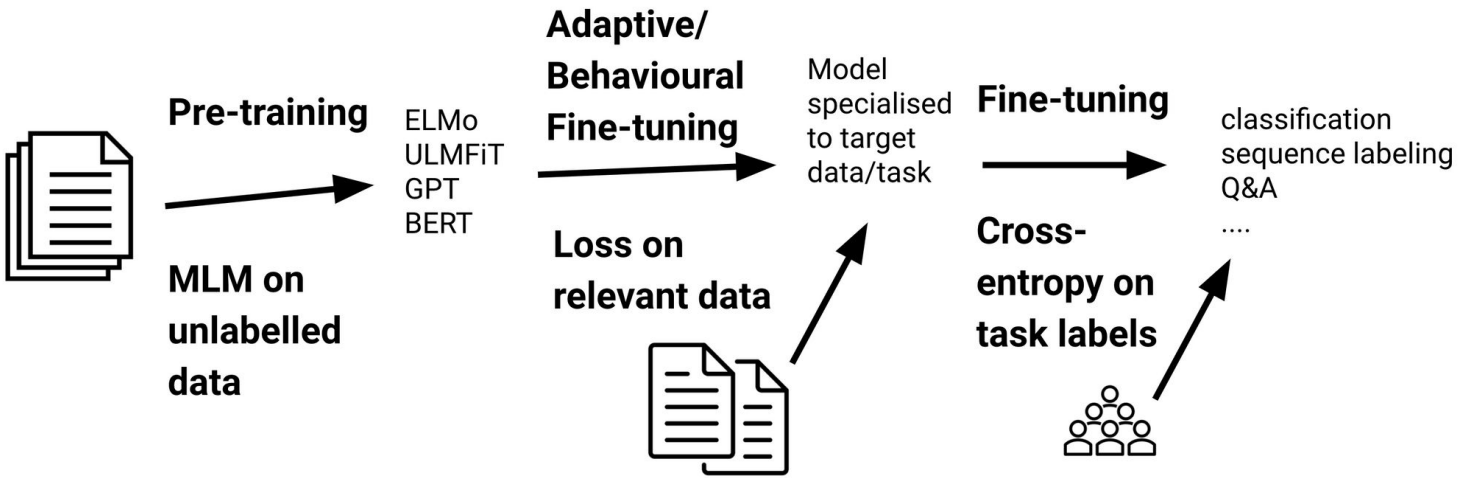
Yes. Pre-training & Fine-tuning is Transfer Learning in my opinion

- The source tasks can be any of **unsupervised/self-supervised/supervised** learning
  - Technically, pre-training is considered more general & task-independent, though

In any case, finding relevant tasks & data is key (domain knowledge!)

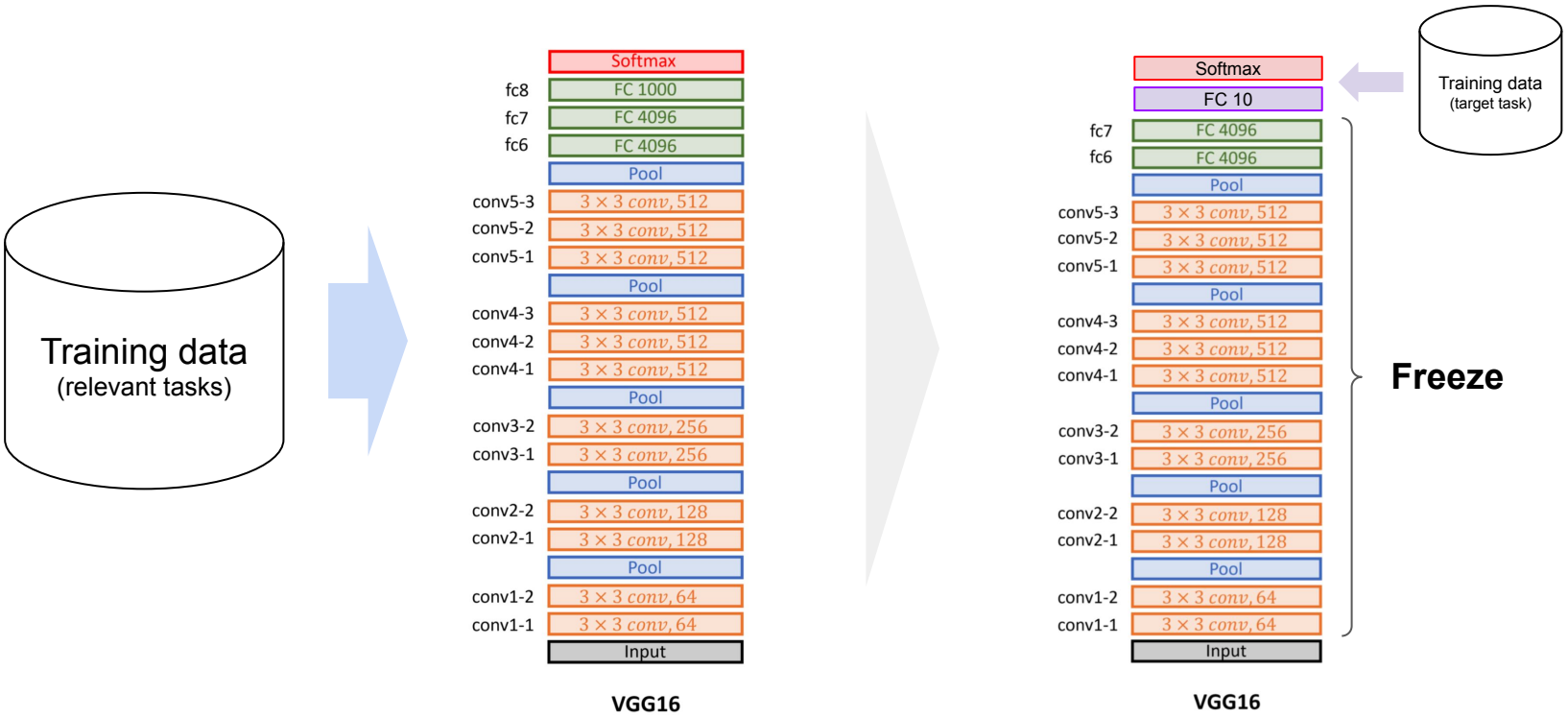
Multi-stage Fine-tuning

- Pre-training → **Adaptive fine-tuning** → Fine-tuning



[Recent Advances in Language Model Fine-tuning - Sebastian Ruder]

Transfer Learning for Computer Vision



Multi-task Learning

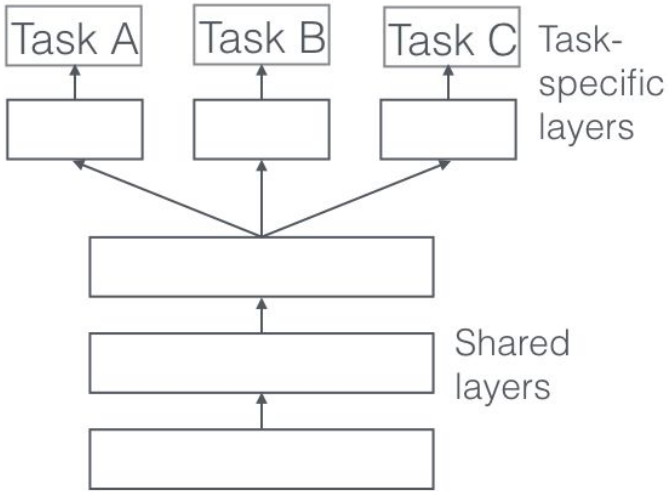
- A machine learning framework that trains a model on multiple (relevant) tasks simultaneously
- Different from Transfer learning, the model is **jointly optimized** on multiple tasks

emotion detection

sentiment analysis

Multi-task Learning: Hard-parameter Sharing

- The most common approach: Attaching an output layer for each task

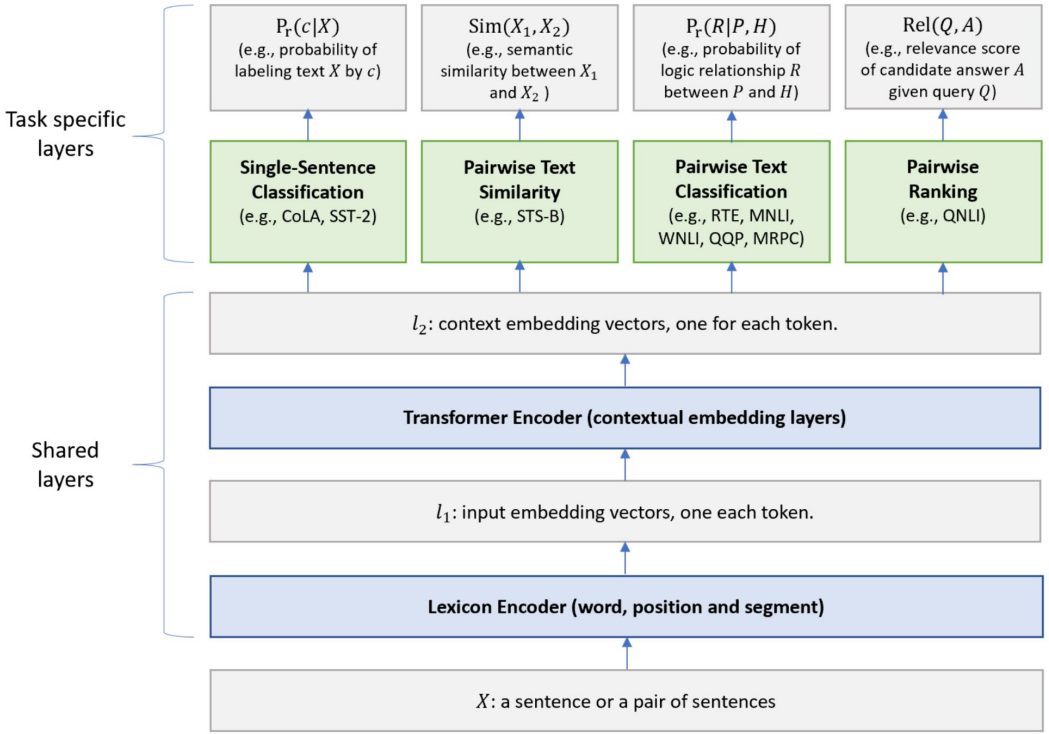


Loss function

$$L = L_1 + \lambda_2 L_2 + \lambda_3 L_3$$

An Overview of Multi-Task Learning for Deep Learning

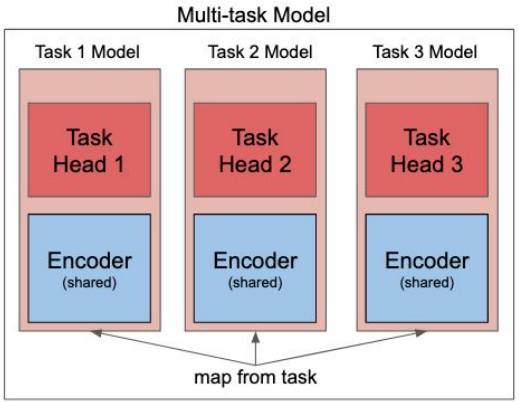
MT-DNN (=~ Multi-task BERT)



[1901.11504] Multi-Task Deep Neural Networks for Natural Language Understanding

Multi-task Learning: Implementation

- It’s easier to create different model objects with *shared parameters*




```
from transformers import (
    AutoModelForSequenceClassification,
    AutoModelForTokenClassification)

model1 = AutoModelForSequenceClassification.from_pretrained("bert-base-uncased")
model2 = AutoModelForTokenClassification.from_pretrained("bert-base-uncased")
model2.bert.embeddings = model1.bert.embeddings
model2.bert.encoder = model1.bert.encoder

...

loss = model1(input_ids, labels=labels1).loss + \
        lambda_2 * model2(input_ids, labels=labels2).loss
```

Multi-task Training with Transformers+NLP

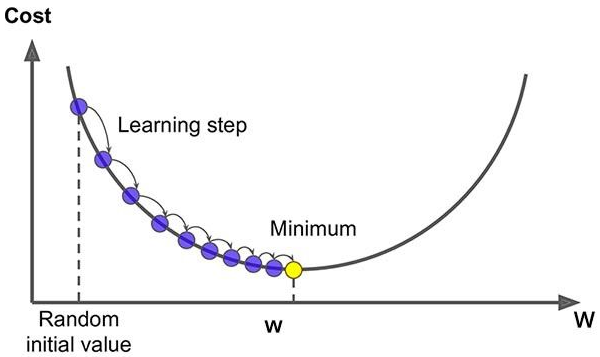
Advanced Optimization Techniques

Optimization strategy

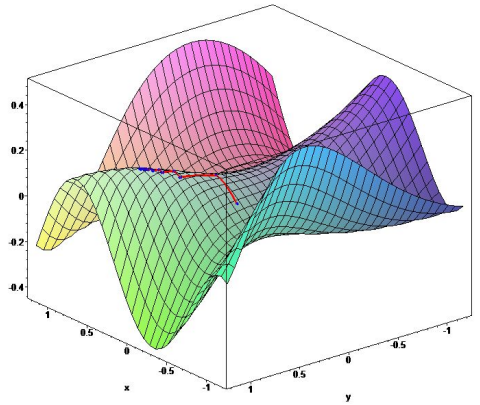
- Basics: Stochastic Gradient Descent
- + alpha
  - Learning rate scheduler
  - Hyper-parameter search
  - Early stopping
  - Hyper-parameter optimization

Stochastic Gradient Descent

- Optimization method based on gradient that is calculated by (randomly chosen) samples



1d parameter space (Convex)



2d parameter space (Non-convex)

$$w^t \leftarrow w^{t-1} - \eta \nabla w$$

Optimization = Gradient (direction) + **Learning rate (Step size)**

Adam (Adaptive moments) (Kingma and Ba 2014)

- SGD + first-order moment
- **+ Second-order moment**

$$m^t \leftarrow \beta_1 m^{t-1} + \left(1-\beta_1\right) \nabla w^t$$

$$v_t \leftarrow \beta_2 v^{t-1} + \left(1-\beta_2\right) \left(\nabla w^t\right)^2$$

- **+ Bias correction**

$$\widehat{m}^t \leftarrow \frac{m^t}{1-\left(\beta_1\right)^t}$$

$$\widehat{v}^t \leftarrow \frac{v^t}{1-\left(\beta_2\right)^t}$$

initial learning rate

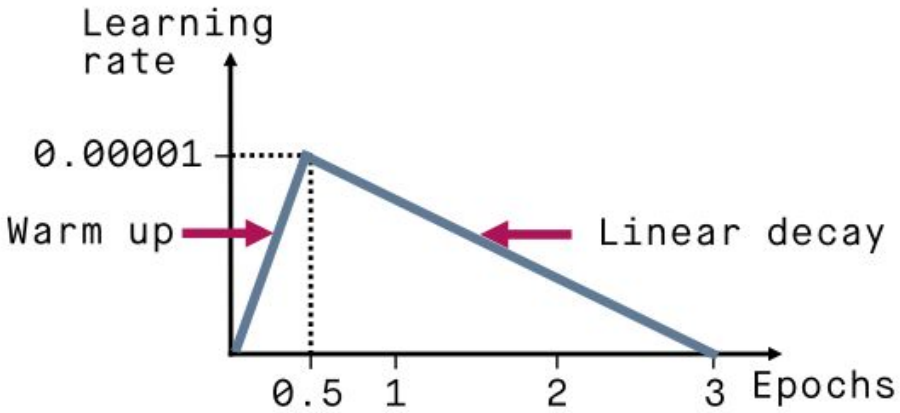
$$w^{t+1} \leftarrow w^t - \frac{\eta}{\sqrt{\widehat{v}^t + \varepsilon}} \widehat{m}^t$$

$$\beta_1 = 0.9,\ \beta_2 = 0.999,\ \varepsilon = 10^{-8}$$

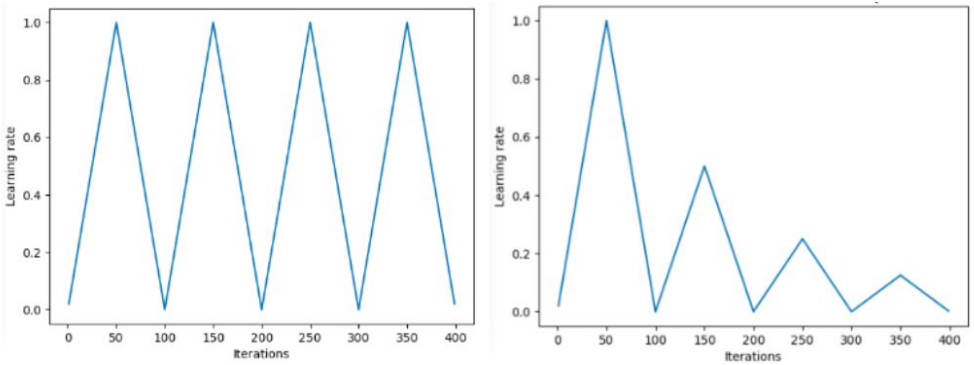
Suggested hyper-parameters

Linear Learning Rate Scheduler

- Warm-up + Linear decay

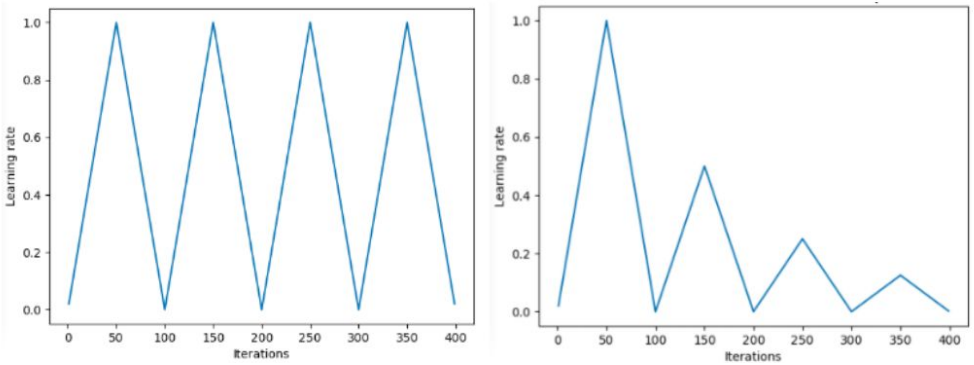


Cyclical Learning Rate Scheduling



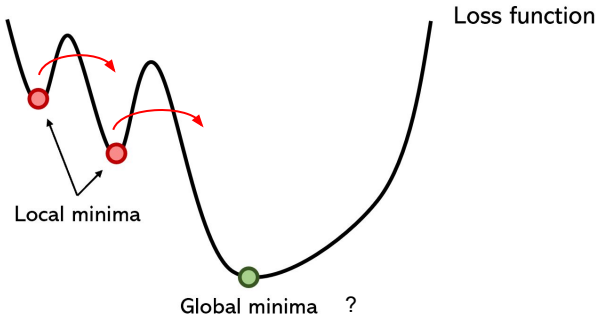
Why do we need this type of learning rate scheduler? 🤔

Cyclical Learning Rate Scheduling



Why do we need this type of learning rate scheduler? 🤔

To find a better local minima



Hyper-parameter Search & Model Selection

- **Grid Search** is a de-facto standard method in the pre-Deep Learning era
- Example
  - Hyper-parameter search for SVM

```
param_grid = [
  {'C': [1, 10, 100, 1000], 'kernel': ['linear']},
  {'C': [1, 10, 100, 1000], 'gamma': [0.001, 0.0001], 'kernel': ['rbf']},
 ]
```

Can you tell the total number of trials?

Hyper-parameter Search & Model Selection

- **Grid Search** is a de-facto standard method in the pre-Deep Learning era
- Example
  - Hyper-parameter search for SVM

```
param_grid = [
  {'C': [1, 10, 100, 1000], 'kernel': ['linear']},
  {'C': [1, 10, 100, 1000], 'gamma': [0.001, 0.0001], 'kernel': ['rbf']},
 ]
```

4 + (4 * 2) = 12

Hyper-parameter Search & Model Selection

- **Grid Search** is a de-facto standard method in the pre-Deep Learning era
- Example
  - Hyper-parameter search for SVM

```
param_grid = [
  {'C': [1, 10, 100, 1000], 'kernel': ['linear']},
  {'C': [1, 10, 100, 1000], 'gamma': [0.001, 0.0001], 'kernel': ['rbf']},
 ]
```

4 + (4 * 2) = 12

We select the model with <u>the best validation performance</u> (e.g., validation accuracy etc.)

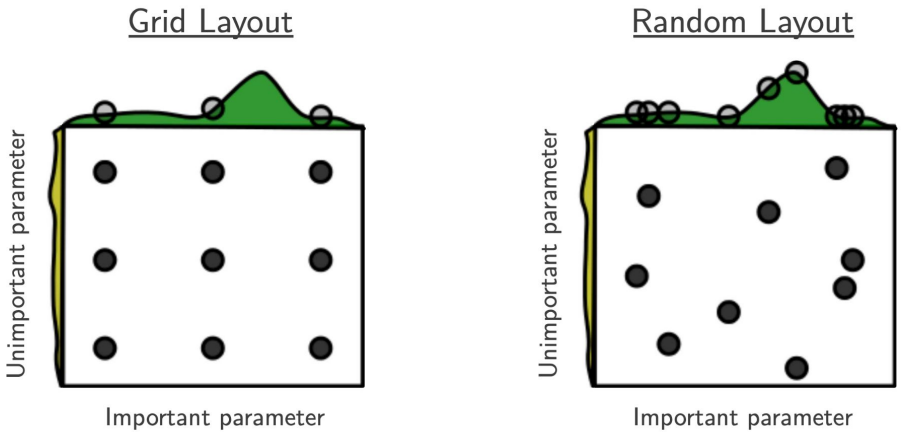
Model Selection for Deep Learning models

- We could still use the same approach but usually don’t
  - A single model training trial takes much longer time

```
param_grid = {“batch”: [8, 16, 32],
              “initial_lr”: [0.001, 0.01, 0.05],
              “epochs”: [5, 10, 15],
              ...}
```

Note: Grid Search vs Random Search

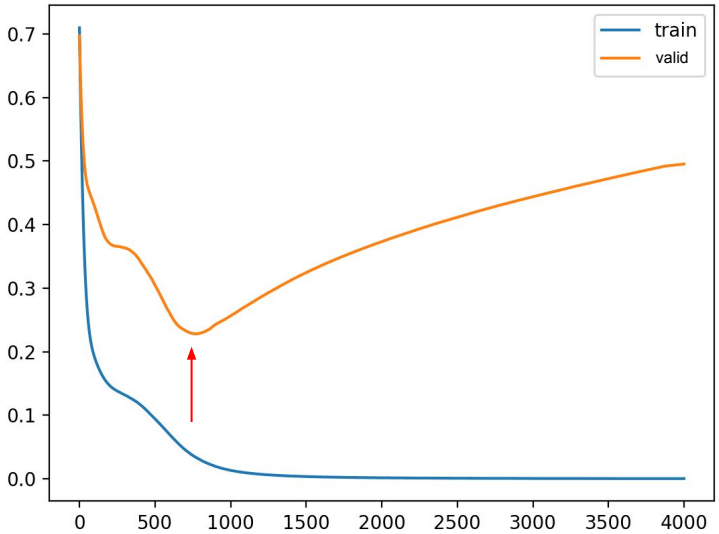
- Random Search is considered a better approach



https://www.jmlr.org/papers/volume13/bergstra12a/bergstra12a.pdf

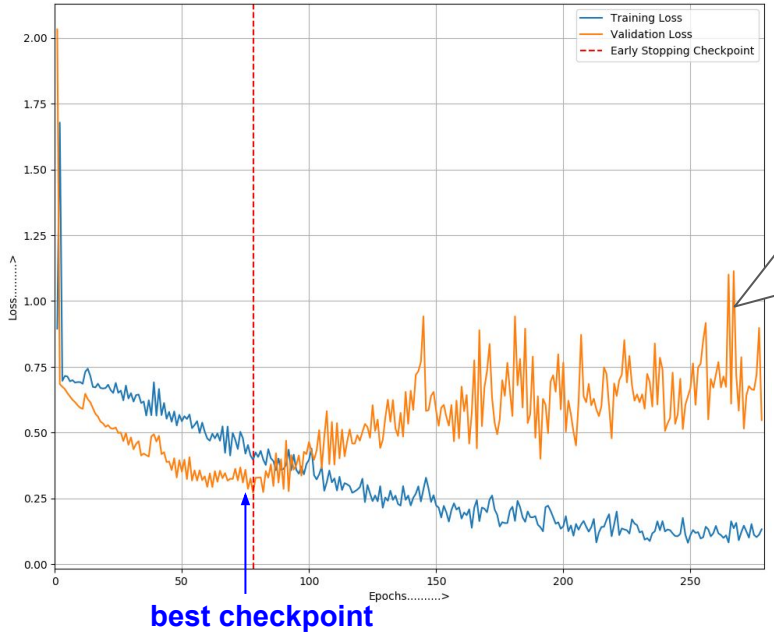
Model Selection: Must-do with Deep Learning Models

- To use the best **checkpoint** chosen by the validation set
  - by saving checkpoint for each epoch (or every T steps)



Early Stopping

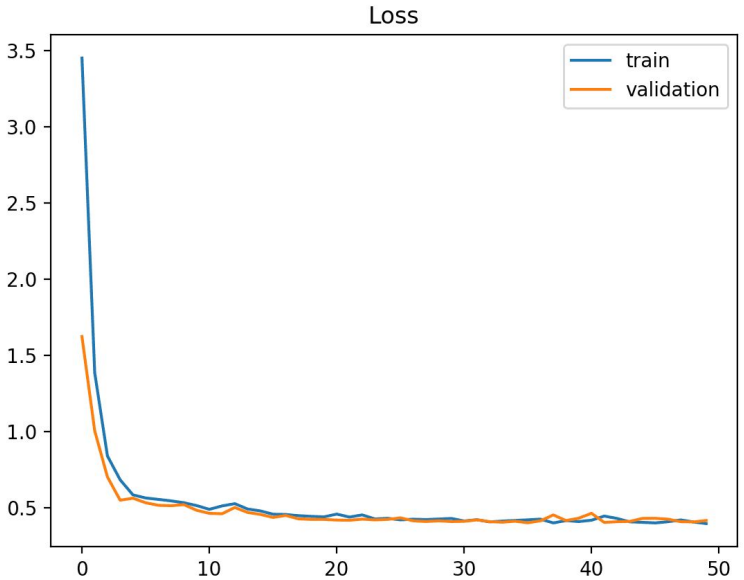
- The training procedure will be terminated if no improvement on **the validation set** is observed for more than *p* epochs (p: patience)



Intuition: The model may keep overfitting to the training data. Why don't you stop training, then?

Reality: Validation loss (can) keep decreasing

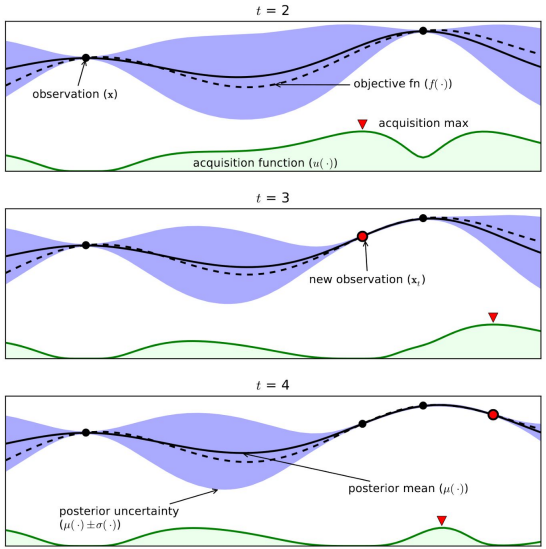
- Furthermore, the validation loss **may not be correlated** with the held-out test accuracy 🤷



Think **and feel**

Hyper-parameter Optimization

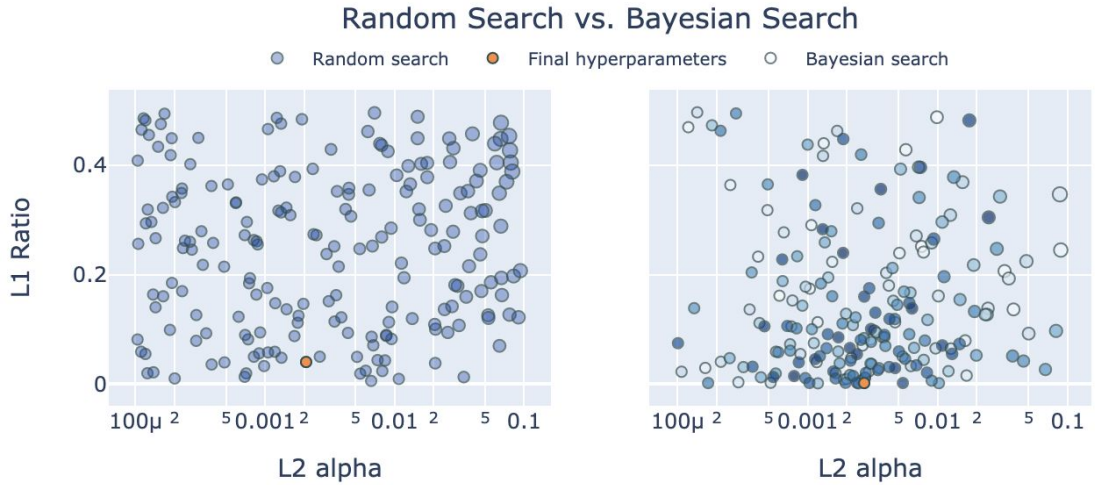
- Evaluation metric is **neither smooth or differentiable** wrt hyper-parameters
- Why don't we approximate a smooth function in the hyper-paramter space?
- → Bayesian Optimization



Shallow Understanding on Bayesian Optimization | by Ramraj Chandradevan

Hyper-parameter Optimization

- Pros :)
  - “smart” Random Search
- Cons :(
  - **Sequential procedure** (= high computational cost)



[Beyond Grid Search: Hypercharge Hyperparameter Tuning for XGBoost]

A Recipe for Optimization

Especially for pre-trained LM fine-tuning

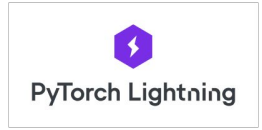
- Rule of thumbs
  - Use AdamW (Adam + weight decay)
  - Use linear learning rate scheduler
  - If possible
    - Use a larger batch size
    - Use a smaller initial learning rate (not too small)
    - Use a larger epoch size

- Further reading
  - [2006.05987] Revisiting Few-sample BERT Fine-tuning (ICLR 2021)
  - [2006.04884] On the Stability of Fine-tuning BERT: Misconceptions, Explanations, and Strong Baselines (ICLR 2021)

Hands-on Session: Trainer

- Google Colab

PyTorch Lightning



- Google Colab (PyTorch Lightning official tutorial)

Summary

- Part I
  - Transformers for Computer Vision
  - Transformer Decoder-only Models (e.g., GPT-2/3)
  - Transformer Encoder-Decoder Models (e.g., BART, T5)

- Part II
  - Sparse Attention
  - Transfer Learning & Fine-tuning
  - Advanced Optimization Techniques

Schedule

- **Fri 11/18:** Finalize the project title and plan
- Tue 11/23: Project presentations (1)
- Tue 11/30: Project presentations (2)
- **Tue 12/7:** Project report deadline