In [1]:
```
%run Latex_macros.ipynb
```

$$ \newcommand{\x}{\mathbf{x}} \newcommand{\tx}{\tilde{\x}} \newcommand{\y}{\mathbf{y}} \newcommand{\b}{\mathbf{b}} \newcommand{\c}{\mathbf{c}} \newcommand{\e}{\mathbf{e}} \newcommand{\z}{\mathbf{z}} \newcommand{\h}{\mathbf{h}} \newcommand{\u}{\mathbf{u}} \newcommand{\v}{\mathbf{v}} \newcommand{\w}{\mathbf{w}} \newcommand{\V}{\mathbf{V}} \newcommand{\W}{\mathbf{W}} \newcommand{\X}{\mathbf{X}} \newcommand{\KL}{\mathbf{KL}} \newcommand{\E}{{\mathbb{E}}} \newcommand{\Reals}{{\mathbb{R}}} \newcommand{\ip}{\mathbf{{(i)}}} % % Test set \newcommand{\xt}{\underline{\x}} \newcommand{\yt}{\underline{\y}} \newcommand{\Xt}{\underline{\X}} \newcommand{\perfm}{\mathcal{P}} % % \ll indexes a layer; we can change the actual letter \newcommand{\ll}{l} \newcommand{\llp}{{(\ll)}} % \newcommand{Thetam}{\Theta_{-0}} % CNN \newcommand{\kernel}{\mathbf{k}} \newcommand{\dim}{d} \newcommand{\idxspatial}{{\text{idx}}} \newcommand{\summaxact}{\text{max}} \newcommand{idxb}{\mathbf{i}} % % % RNN % \tt indexes a time step \newcommand{\tt}{t} \newcommand{\tp}{{(\tt)}} % % % LSTM \newcommand{\g}{\mathbf{g}} \newcommand{\remember}{\mathbf{remember}} \newcommand{\save}{\mathbf{save}} \newcommand{\focus}{\mathbf{focus}} % % % NLP \newcommand{\Vocab}{\mathbf{V}} \newcommand{\v}{\mathbf{v}} \newcommand{\offset}{o} \newcommand{\o}{o} \newcommand{\Emb}{\mathbf{E}} % % \newcommand{\loss}{\mathcal{L}} \newcommand{\cost}{\mathcal{L}} % % \newcommand{\pdata}{p_\text{data}} \newcommand{\pmodel}{p_\text{model}} % % SVM \newcommand{\margin}{{\mathbb{m}}} \newcommand{\lmk}{\boldsymbol{\ell}} % % % LLM Reasoning \newcommand{\rat}{\mathbf{r}} \newcommand{\model}{\mathcal{M}} \newcommand{\bthink}{\text{}} \newcommand{\ethink}{\text{}} % % % Functions with arguments \def\xsy#1#2{#1^#2} \def\rand#1{\tilde{#1}} \def\randx{\rand{\x}} \def\randy{\rand{\y}} \def\trans#1{\dot{#1}}

\def\transx{\trans{\x}} \def\transy{\trans{\y}} % \def\argmax#1{\underset{#1} {\operatorname{argmax}} } \def\argmin#1{\underset{#1} {\operatorname{argmin}} } \def\max#1{\underset{#1} {\operatorname{max}} } \def\min#1{\underset{#1} {\operatorname{min}} } % \def\pr#1{\mathcal{p}(#1)} \def\prc#1#2{\mathcal{p}(#1 \; | \; #2)} \def\cnt#1{\mathcal{count}_{#1}} \def\node#1{\mathbb{#1}} % \def\loc#1{{\text{##} {#1}}} % \def\OrderOf#1{\mathcal{O}\left({#1} \right)} % % Expectation operator \def\Exp#1{\underset{#1} {\operatorname{\mathbb{E}}} } % % VAE \def\prs#1#2{\mathcal{p}_{#2}(#1)} \def\qr#1{\mathcal{q}(#1)} \def\qrs#1#2{\mathcal{q}_{#2}(#1)} % % Reinforcement learning \newcommand{\Actions}{{\mathcal{A}}} \newcommand{\actseq}{A} \newcommand{\act}{a} \newcommand{\States}{{\mathcal{S}}} \newcommand{\stateseq}{S} \newcommand{\state}{s} \newcommand{\Rewards}{{\mathcal{R}}} \newcommand{\rewseq}{R} \newcommand{\rew}{r} \newcommand{\transp}{P} \newcommand{\statevalfun}{v} \newcommand{\actvalfun}{q} \newcommand{\disc}{\gamma} % % \newcommand{\floor}[1]{\left\lfloor #1 \right\rfloor} \newcommand{\ceil}[1]{\left\lceil #1 \right\rceil} % % $$

Classical Machine Learning

Week 0

Plan

- Setting up your learning and programming environment

Getting started

- Setting up your ML environment (Setup_NYU.ipynb)
    - Choosing an ML environment (Choosing_an_ML_Environment_NYU.ipynb)
- Quick intro to the tools (Getting_Started.ipynb)

Week 1

Plan

We give a brief introduction to the course.

We then present the key concepts that form the basis for this course

- For some: this will be review

Intro to Advanced Course

- Introduction to Advanced Course (Intro_Advanced.ipynb)

Using an AI Assistant as a Personal Tutor

Knowledge of Deep Learning is a prerequisite for this course.

For those of you who need a review, we will do so at a **very rapid pace and abbreviated manner**.

But using AI Assistants (e.g., ChatGPT) can be a great resource for getting you up to speed.

The key is to using them as a personal tutor. Some advice

- describe the role you want them to play (e.g., Professor, practitioner)
    - this sets the level for depth of knowledge it will convey
- describe your level of knowledge (Graduate student, undergraduate, hacker)
    - this sets the level of complexity for the responses it provides
- Treat it as a tutor
    - ask for a concept to be explained, at varying levels of depth
    - ask follow-up questions until you get what you need

Here (https://www.perplexity.ai/search/you-are-an-expert-in-deep-lear-qPHF81eAScGyAJogE5idCw?0=) is an example of such a conversation using Perplexity as the AI Assistant

- you may use any Assistant that you prefer: they are very similar in capabilities

Suggested reading

HuggingFace course (https://huggingface.co/course)

- you are well-advised to follow this material over the next 3 weeks in preparation for the Course Project

Review/Preview of concepts from Intro Course (very abbreviated)

Here is a *quick reference* of key concepts/notations from the Intro course

- For some: it will be a review, for others: it will be a preview.
- We will devote a sub-module of this lecture to elaborate on each topic in slightly more depth.
    - For a more detailed explanation: please refer to the material from the Intro course (repo (https://github.com/kenperry-public/ML_Spring_2023))
- Review and Preview (Review_Advanced.ipynb)

You may want to run your code on Google Colab in order to take advantage of powerful GPU's.

Here are some useful tips:

Google Colab tricks (Colab_practical.ipynb)

Transformers: Review

Preview

There is lots of interest in Large Language Models (e.g., ChatGPT). These are based on an architecture called the Transformer. We will introduce the Transformer and demonstrate some amazing results achieved by using Transformers to create Large Language Models.

Attention is a mechanism that is a core part of the Transformer. We will begin by first introducing Attention.

We will then take a detour and study the Functional model architecture of Keras. Unlike the Sequential model, which is an ordered sequence of Layers, the organization of blocks in a Functional model is more general. The Advanced architectures (e.g., the Transformer) are built using the Functional model.

Once we understand the technical prerequisites, we will examine the code for the Transformer.

Transformers: Review (Review_Transformer.ipynb)

Suggested reading

- Attention
    - Attention is all you need (https://arxiv.org/pdf/1706.03762.pdf)
- Transfer Learning
    - Sebastian Ruder: Transfer Learning (https://ruder.io/transfer-learning/)
- HuggingFace course
    - Transformers: concepts (https://huggingface.co/learn/nlp-course/chapter1/4?fw=pt)

Further reading

- Attention
    - Neural Machine Translation by Jointly Learning To Align and Translate

The remaining "reviews" will be omitted in class

- we will reference them in passing when dealing with the related topic in future lectures

Transformer: flavors (Transformer.ipynb#Transformer-variants)

Attention: in depth

- Implementing Attention (Attention_Lookup.ipynb)

Transfer Learning: Review (Review_TransferLearning.ipynb)

Natural Language Processing: Review (Review_NLP.ipynb)

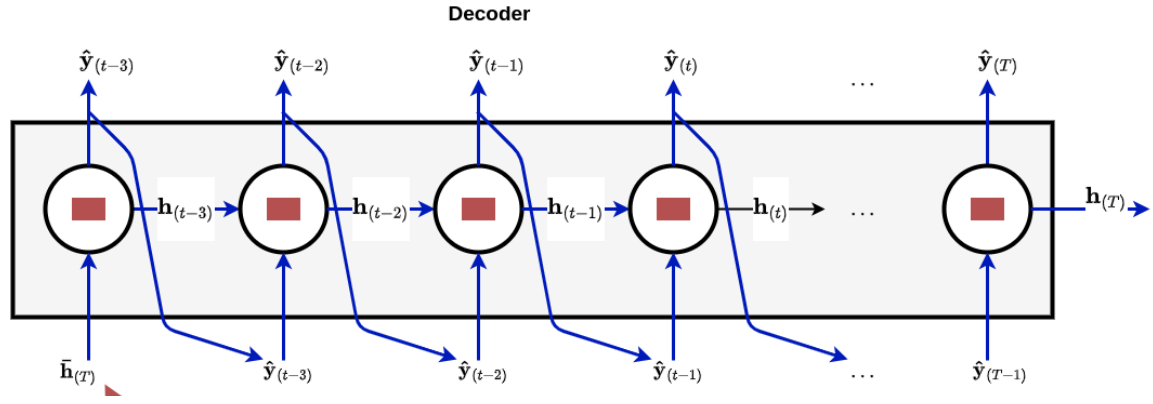
LLM: Review (Review_LLM.ipynb)

Week 2: Technical tools/background

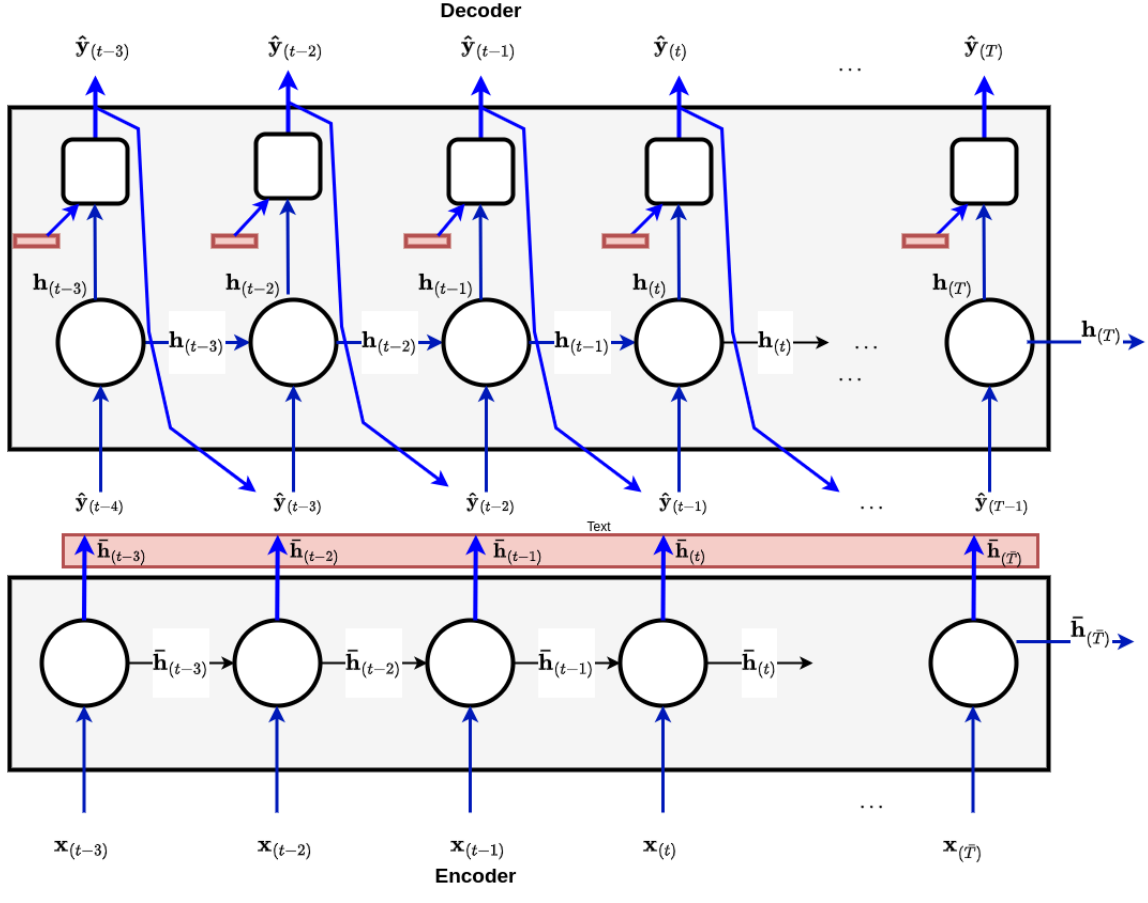
Review: wrap-up

From RNN to Transformer: Attention is all you need

RNN Encoder/Decoder without Attention
Bottleneck


Decoder
$\hat{\mathbf{y}}_{(t-3)}$
$\hat{\mathbf{y}}_{(t-2)}$
$\hat{\mathbf{y}}_{(t-1)}$
$\hat{\mathbf{y}}_{(t)}$
$\hat{\mathbf{y}}_{(T)}$
$\mathbf{h}_{(t-3)}$
$\mathbf{h}_{(t-2)}$
$\mathbf{h}_{(t-1)}$
$\mathbf{h}_{(t)}$
$\mathbf{h}_{(T)}$
$\bar{\mathbf{h}}_{(T)}$
$\hat{\mathbf{y}}_{(t-3)}$
$\hat{\mathbf{y}}_{(t-2)}$
$\hat{\mathbf{y}}_{(t-1)}$
$\hat{\mathbf{y}}_{(T-1)}$

RNN Encoder/Decoder with Attention



Subject: Professor Perry

Pronoun: he

Natural Language Processing: Review (Review_NLP.ipynb)

LLM: Review (Review_LLM.ipynb)

Suggested reading

HuggingFace course (https://huggingface.co/course)

- you are well-advised to follow this material over the next 3 weeks in preparation for the Course Project

Functional Models

Plan

Enough theory (for the moment) !

The Transformer (whose theory we have presented) is built from plain Keras.

Our goal is to dig into the **code** for the Transformer so that you too will learn how to build advanced models.

Before we can do this, we must

- go beyond the Sequential model of Keras: introduction to the Functional model
- understand more "advanced" features of Keras: customomizing layers, training loops, loss functions
- The Datasets API

Basics

We start with the basics of Functional models, and will give a coding example of such a model in Finance.

- Functional API (Functional_Models.ipynb)

Suggested reading

- Keras docs
    - Functional API (https://keras.io/guides/functional_api/)
    - Making new layers and models via sub-classing (https://keras.io/guides/making_new_layers_and_models_via_subclassing/)

Advanced Keras (Deeper dive)

We will not cover this notebook (Keras_Advanced.ipynb) in class

- most of the material will be introduced as part of our study of different interesting models
- but this notebook is one convenient place to see them all
- consider it as a **reference** that collects multiple techniques in one place

Deeper dives

If you *really* want to dig into the micro-details of TensorFlow, here are some important subtleties

- Computation Graphs (Computation_Graphs.ipynb)
- Eager vs Graph Execution (TF_Graph.ipynb)

Functional Model Code: A Functional model in Finance: "Factor model"

We illustrate the basic features of Functional models with an example

- does not use the additional techniques of the next section (Advanced Keras)

Autoencoders for Conditional Risk Factors (Autoencoder_for_conditional_risk_factors.ipynb)

- code (https://github.com/stefan-jansen/machine-learning-for-trading/blob/main/20_autoencoders_for_conditional_risk_factors/06_conditional_a

Suggested reading

- Autoencoder asset pricing models (https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3335536)

Putting it all together: Code: the Transformer (continued)

Plan

With the base of advanced Keras under our belts, it's time to understand the Transformer, in code

We will examine the code in the excellent [TensorFlow tutorial on the Transformer (https://www.tensorflow.org/text/tutorials/transformer)]

- more in-depth than our presentation
- more background

The tutorial is especially recommended for those without the basics of the Transformer from my Intro course

- [The Transformer: Understanding the Pieces (Transformer_Understanding_the_Pieces.ipynb)]
- [The Transformer: Code (Transformer_code.ipynb)]
- [Implementing Attention: detail (Implementing_Attention.ipynb)]
- [Choosing a Transformer architecture (Transformer_Choosing_a_PreTrained_Model.ipynb)]

Suggested reading

There is an excellent tutorial on Attention and the Transformer which I recommend:

- Tensorflow tutorial: Neural machine translation with a Transformer and Keras (https://www.tensorflow.org/text/tutorials/transformer)

Deeper dive

- Implementing Attention (Attention_Lookup.ipynb)
- Residual connections (RNN_Residual_Networks.ipynb)

Datasets: Big data in small memory

Plan

We continue our exploration of the Functional API in Keras.

We will spend some time examining the code for the Transformer.

We will also introduce the TensorFlow Dataset (TFDS) API, a way to consume large datasets using a limited amount of memory.

Plan

Last piece of technical info to enable the project

- TensorFlow Dataset (TF_Data_API.ipynb)

Background

- Python generators (Generators.ipynb)

Notebooks

- Dataset API: play around (TFDatasets_play_v1.ipynb)

Week 3: Fine-tuning a pre-trained model; The HuggingFace ”ecosystem”

Beyond Transfer Learning: Fine-tuning a pre-trained model

Plan

We begin the "technical" part of the course: the programming tools that will enable the Course Project.

We introduce "Modern Transfer Learning": using model hubs.

The hub we will use for the final project: HuggingFace

- illustrate how to fine-tune a pre-trained model
- quick Intro to HF
    - best way to learn: through the course !
    - uses Datasets
        - will introduce later
    - PyTorch version (uses Trainer); we will focus on Tensorflow/Keras version

HuggingFace Transformers course

The best way to understand and use modern Transfer Learning is via the HuggingFace course (https://huggingface.co/course).

You will learn

- about the Transformer
- how to use HuggingFace's tools for NLP (e.g., Tokenizers)
- how to perform common NLP tasks
    - especially with Transformers
- how to fine-tune a pre-trained model
- how to use the HuggingFace dataset API

All of this will be invaluable for the Course Project.

- does not have to be done using HuggingFace
- but using at least parts of it will make your task easier

- HuggingFace intro (Transfer_Learning_HF.ipynb)
    - linked notebook: Using a pretrained Sequence Classifier (HF_quick_intro_to_models.ipynb) **(local machine)**
    - linked notebook: Using a pretrained Sequence Classifier (https://colab.research.google.com/github/kenperry-public/ML_Advanced_Fall_2025/blob/master/HF_quick_intro_to_models.ip **(Google Colab)**
        - Examining a model

Suggested reading

HuggingFace course (https://huggingface.co/course)

Fine Tuning at low cost

- you are well-advised to follow this material over the next 3 weeks in preparation

Parameter Efficient Transfer Learning

Transfer learning may be the "future of Deep Learning" in that we can adapt models (that are too big for us to train on our own) to our own tasks.

But Fine-Tuning a Pre-Trained model involves training a lot of parameters when the base model is large.

This may be difficult for a variety of reasons.

Can we adapt a Pre-Trained base model to a new task *without* training a large number of parameters ?

- Parameter Efficient Transfer Learning (ParameterEfficient_TransferLearning.ipynb)

Suggested reading

- Parameter Efficient Transfer Learning - article (https://lightning.ai/pages/community/article/understanding-llama-adapters/)
- LoRA:Low Rank Adaptation of Large Language Models (https://arxiv.org/pdf/2106.09685.pdf)
- Adapters
    - Parameter Efficient Transfer Learning for NLP (https://arxiv.org/pdf/1902.00751.pdf)
    - LLM Adapters (https://arxiv.org/pdf/2304.01933.pdf)

Additional reading

- Intrinsic Dimensionality Explains the Effectiveness of Language Model Fine-Tuning (https://arxiv.org/abs/2012.13255)
- LoRA Learns Less and Forgets Less (https://arxiv.org/pdf/2405.09673)

Fine Tuning by Proxy

Can we fine-tune a large model

- **without** adapting its weights

Fine Tuning by Prompt Engineering

Can we adapt a base LLM to solve a new Target task just by changing the prompt ?

Surprisingly: yes !

- Fine Tuning by Prompt Tuning (Prompt_Engineering_Tuning.ipynb)
- Few-shot and Zero shot learning are included in the prompting techniques studied. Why do they work ?
    - In Context Learning: Theory (In_Context_Learning_Theory.ipynb)

Fine-Tuning "without** fine-tuning" learning a new task from exemplars

- In Context Learning (Review_LLM.ipynb#Universal-API/In-context-Learning)

Weel 4: Future of LLM's

Transformers: Scaling

We now have the capabilities to build models with extremely large number of weights. Is it possible to have too many weights ?

Yes: weights, number of training examples and compute capacity combine to determine the performance of a model.

There is an empirical result that suggests that in order to take advantage of GPT-3's use of 175 billion weights

- 1000 times more compute is required than what was used
- 10 times more training examples is required compared to what was used

How large should my Transformer be ? (Transformers_Scaling.ipynb)

Suggested reading

- Scaling laws (https://arxiv.org/pdf/2001.08361.pdf)

Further reading

- Inference budget
    - Transformer Inference Arithmetic (https://kipp.ly/transformer-inference-arithmetic/)
    - LLaMA: Open and Efficient Foundation Language Models (https://arxiv.org/pdf/2302.13971.pdf)

Additional Deep Learning resources

Here are some resources that I have found very useful.

Some of them are very nitty-gritty, deep-in-the-weeds (even the "introductory" courses)

- For example: let's make believe PyTorch (or Keras/TensorFlow) didn't exists; let's invent Deep Learning without it !
    - You will gain a deeper appreciation and understanding by re-inventing that which you take for granted

Andrej Karpathy course: Neural Networks, Zero to Hero (https://karpathy.ai/zero-to-hero.html)

- PyTorch
- Introductory, but at a very deep level of understanding
    - you will get very deep into the weeds (hand-coding gradients !) but develop a deeper appreciation

fast.ai

`fast.ai` is a web-site with free courses from Jeremy Howard.

- PyTorch
- Introductory and courses "for coders"
- Same courses offered every few years, but sufficiently different so as to make it worthwhile to repeat the course !
    - Practical Deep Learning (https://course.fast.ai/)
    - Stable diffusion (https://course.fast.ai/Lessons/part2.html)
        - Very detailed, nitty-gritty details (like Karpathy) that will give you a deeper appreciation

Stefan Jansen: Machine Learning for Trading (https://github.com/stefan-jansen/machine-learning-for-trading)

An excellent github repo with notebooks

- using Deep Learning for trading
- Keras
- many notebooks are cleaner implementations of published models

Assignments

Your assignments should follow the Assignment Guidelines (assignments/Assignment_Guidelines.ipynb)

In [2]:
```
print("Done")
```

```
Done
```