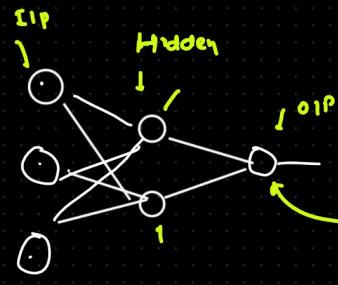
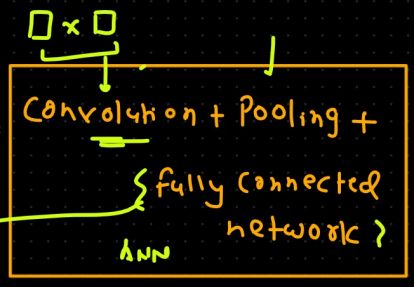
GEN-AI History

1. Language Modelling
2. RNN → Transformer
3. LLM
4. BERT
5. GPT
7. ChatGPT training
8. Transfer Learning & Fine tuning NLP

DL Type of NN [NN is a fundamental unit of DL]

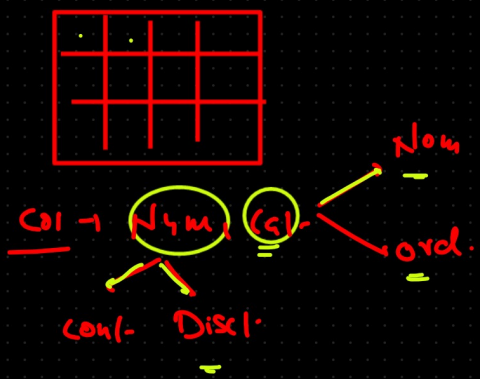
1. ANN
2. CNN
3. RNN



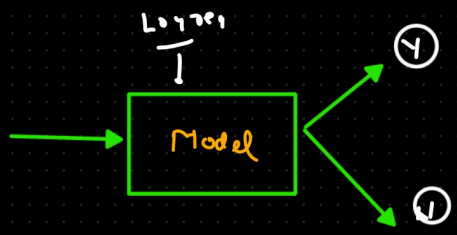


ANN + Feedback loop

- Structure Data
- Image / videos
- Sequence data
  text / Audio



Discriminative Model v/s Creative Model



1. Specific task
2. Limited data

1. genral task
2. huge data

Creative Model type

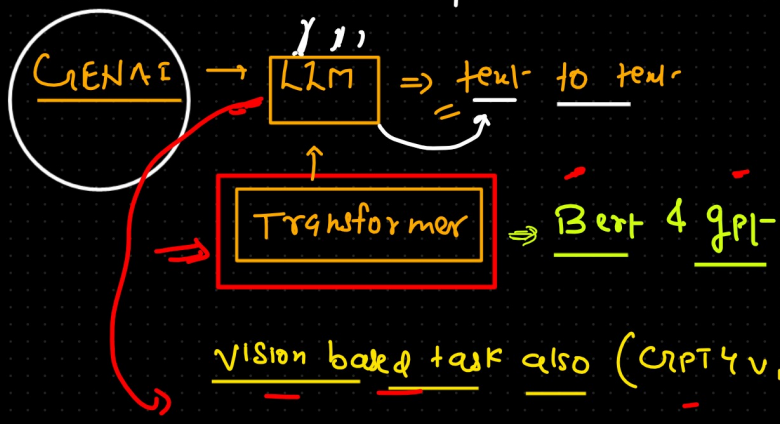
1. image to image
2. text to text

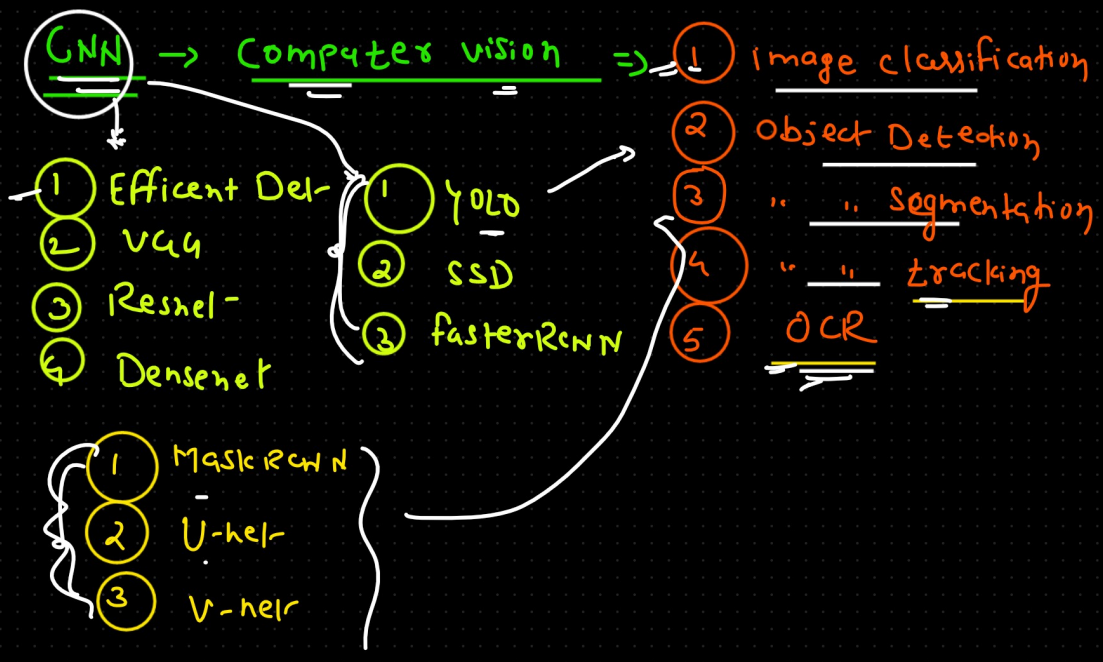
} homogenous Model

3. image to text
4. text to image

} hetrogenous Model or (Multimodal) or (Diffusion model)

Visual + Seq



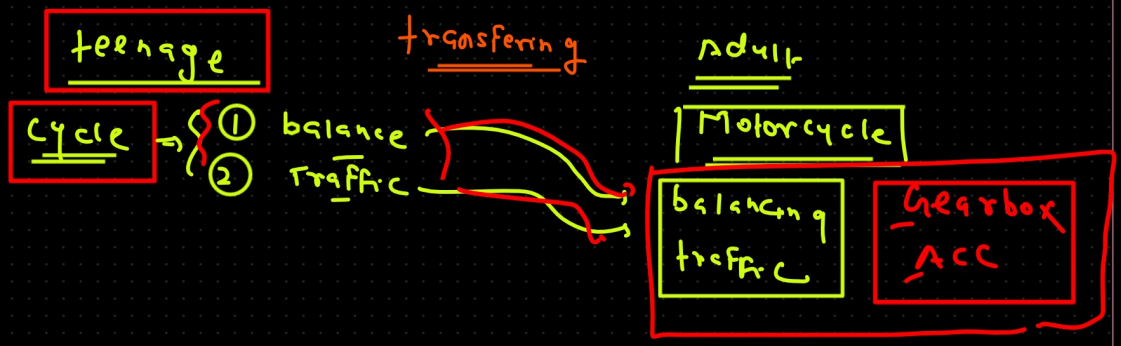


Transfer Learning & Finetuning

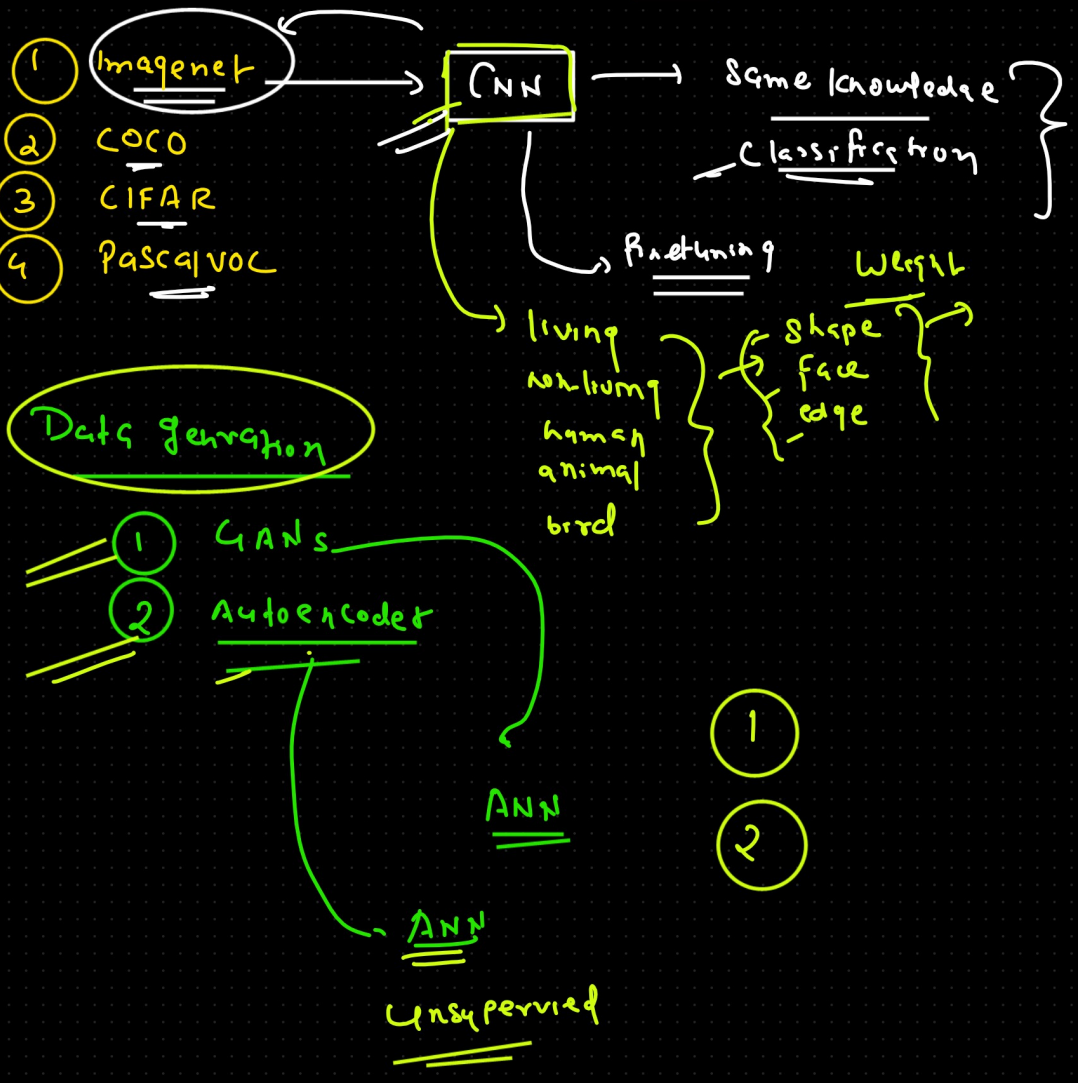
Use the Previous Learning for the future task

Modified Previous Learning According to requirement

Ex:-



Transfer learning & Finetuning (CNN) (Computer vision)

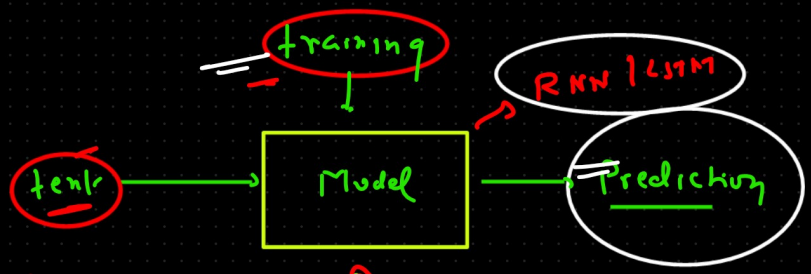


NLP ⇒ Sequence data, text data

1. Summarization
2. Generation (word, sentence)
3. translation (
4. classification

Model

1. RNN → 1987
2. LSTM → 1997
3. GRU → 2014

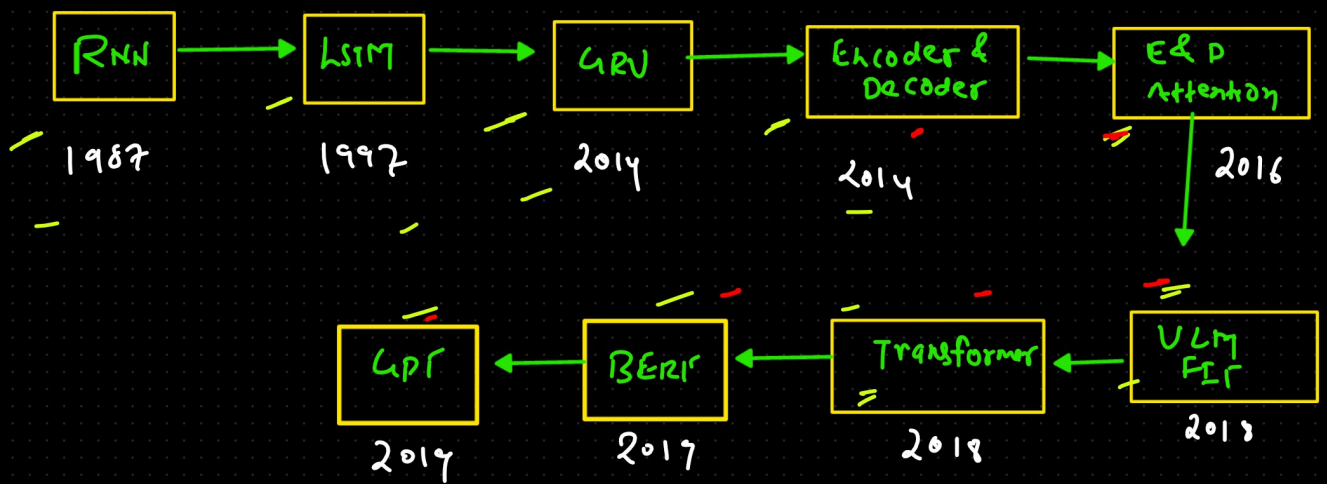


text	target
Sunny is good	+ve
sunny is bad	-ve

Supervised

Language Modelling

Complete timeline of the evolution of LLM



① Mapping (Seq to Seq mapping)

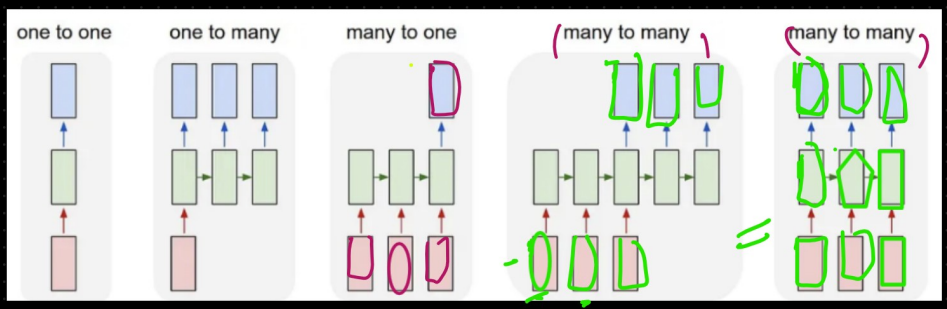


image classification

Image caption (CNN+RNN)

Sentiment (RNN, LSTM)

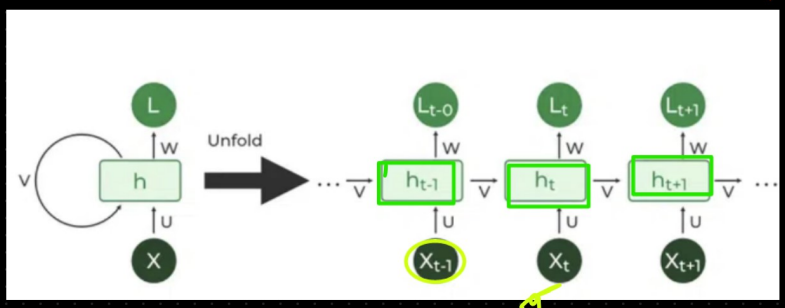
{ language translation, NER, POS

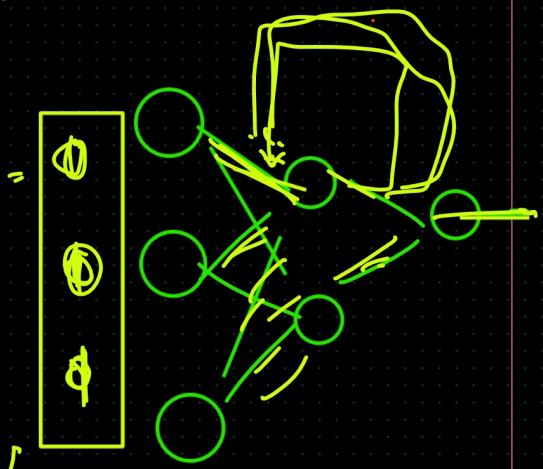
→ Same length

Diff length

② RNN

{ Switch off the fan





I am Sunny

10 · 010 001

Working ⇒ time stamp

Sequential Process

Advantage ⇒ Seq

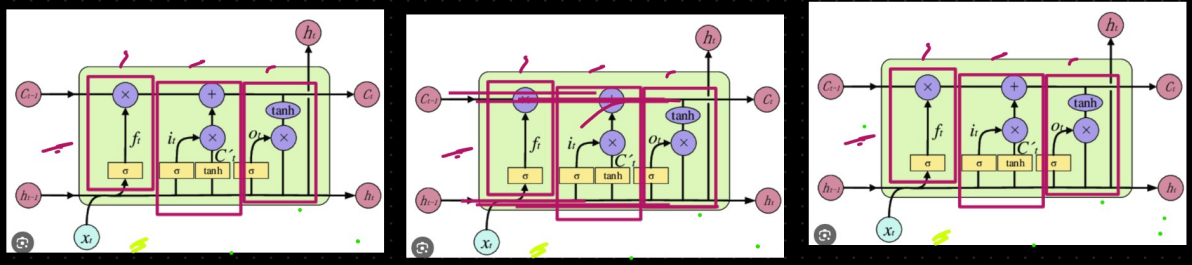
Disadvantage ⇒ (long context)

optimization

① vanishing gradient

② exploding ..

(3) LSTM (Long short term memory)



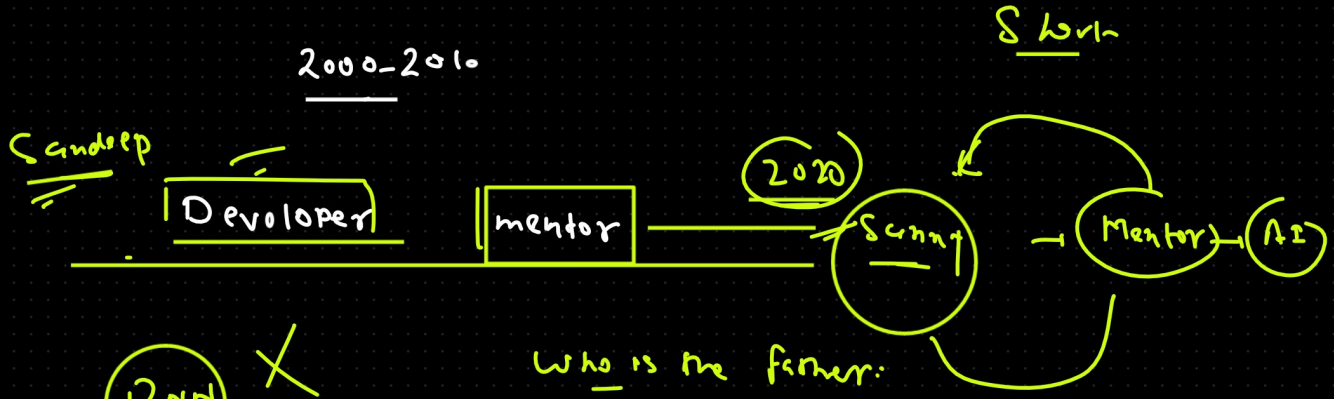
t=1 t=2 t=3

Little longer Sentence

= long - short

Sunny, Sandeep

2000-2010

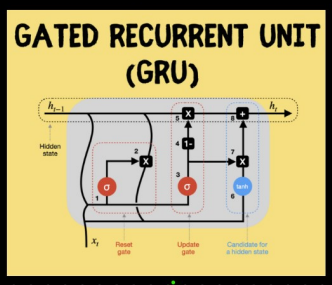


RNN X

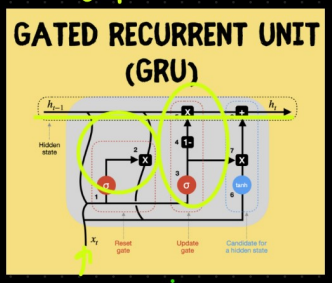
LSTM

4 GRU (Gated Recurrent Unit)

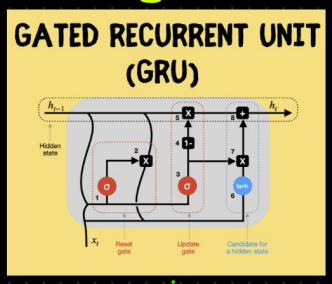
t=0



t=1



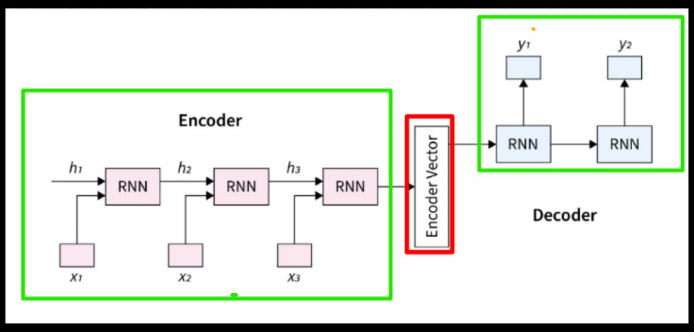
t=2



Feature	LSTM	GRU
Architecture Complexity	More complex with three gates (input, forget, output) and a separate memory cell.	Simpler with two gates (update, reset) and a combined hidden state/memory cell.
Memory Handling	Separate memory cell explicitly manages what information to keep or discard.	Combines hidden state and memory cell into a single state vector.
Gating Mechanism	Explicit forget, input, and output gates for controlling information flow.	Single update gate for combining past and new information, and a reset gate.
Parameter Count	Higher due to the additional parameters from the complex architecture.	Lower, as GRUs have fewer parameters, making them computationally more efficient.
Training Convergence	May require more time to converge during training due to the complexity.	Often converges faster during training, making them easier to train in some scenarios.

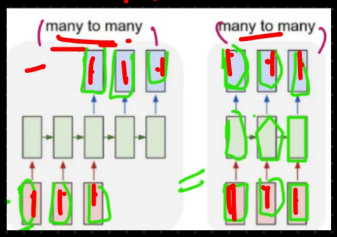
Choosing between LSTM and GRU depends on the specific task, dataset, and computational resources. LSTMs might be more effective in capturing longer dependencies due to their more intricate structure, but GRUs often require fewer parameters and may be computationally more efficient in certain scenarios. It's common practice to experiment with both architectures to determine which one performs better for a particular use case.

(6) Encoder & Decoder



1. Encoder
2. Decoder
3. RNN | LSTM | GRU ⟶ LSTM ⇒ Bi-Directional
4. Encoder ⟶ Content vector ⟶ decoder

Why ⇒ Seq to seq mapping (many to many) - (Lang. translator)



I am sunny

eg:- Language translation (Synchronized) { मैं ठीक हूँ / I am sunny }
(Same length, small sentance)

Longer sentence, Different length { मुझे लगता है कि मैं switch off the fan }
(Asynchronize)

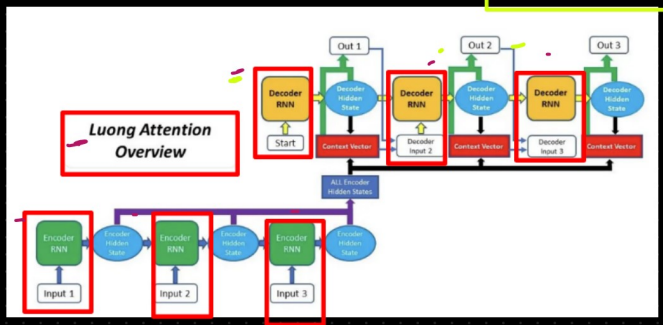
Research only → 30-35 words

It was fine otherwise giving the issue

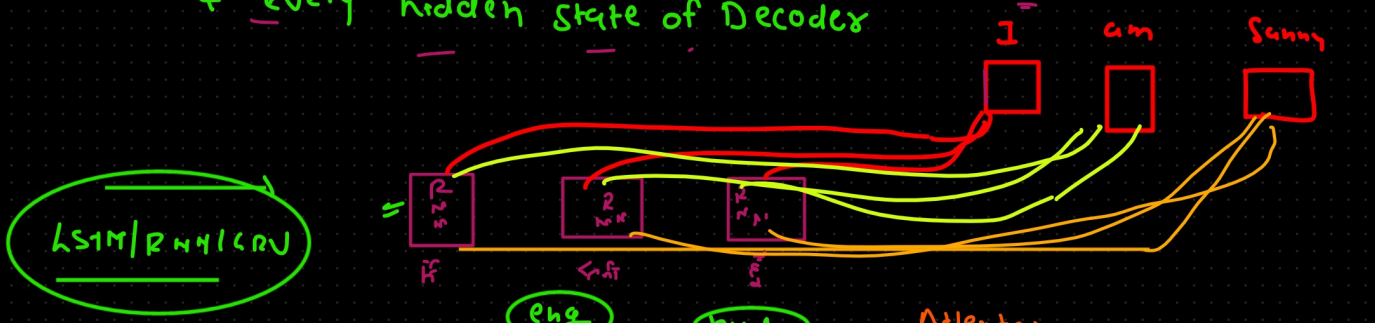
Who was using it ⇒ Google was using it in their earlier days

What was the solution ⇒ E&D with Attention

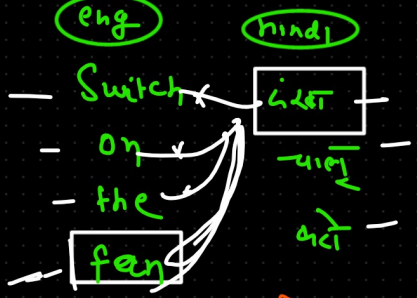
(7) Encoder & Decoder with attention



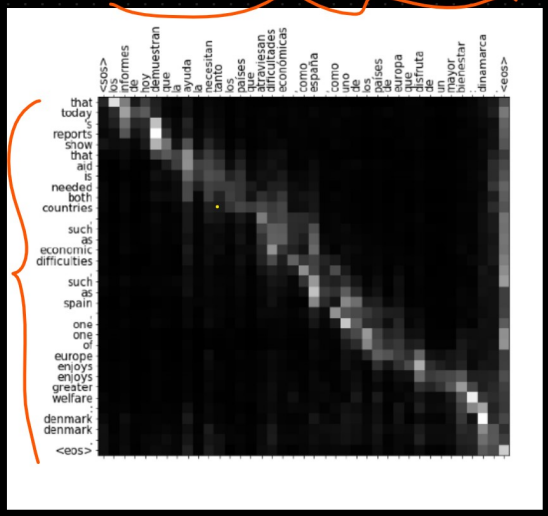
1. Not only single Context vector
2. Connecting each & every hidden state of encoder with each & every hidden state of Decoder



Ex:-



Attention



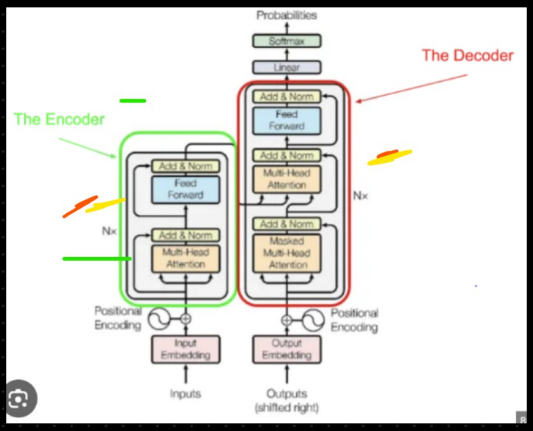
Heat map

Intensity of mapping

More white → strong relation Black → No-relation

less white → weak Relation

8 Transformer (Attention all you need)



1. Remove ~~LSTM/RNN~~ ← GRO
2. Self attention → (Multihead attention)
3. Positional encoding
4. Parallel processing
5. Normalization
6. Artificial Neural Network
7. Skip Connection

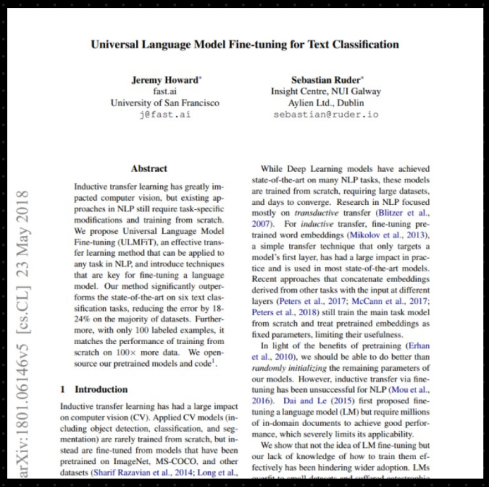
8 Application beyond NLP

thing required

1. Data ⇒ huge data
2. Hardware → GPU, Distributed
3. Time

9 ULMFIT ⇒

1. Universal Lang. modelling
2. Fine tuning for text-classification



Universal Language Model Fine-tuning for Text Classification

Jeremy Howard*
fast.ai
University of San Francisco
j@fast.ai

Sebastian Ruder*
Insight Centre, NUI Galway
Aylien Ltd., Dublin
sebastian@ruder.io

Abstract

Inductive transfer learning has greatly impacted computer vision, but existing approaches in NLP still require task-specific modifications and training from scratch. We propose Universal Language Model Fine-tuning (ULMFiT), an effective transfer learning method that can be applied to any task in NLP, and introduce techniques that are key for fine-tuning a language model. Our method significantly outperforms the state-of-the-art on six text classification tasks, reducing the error by 18-24% on the majority of datasets. Furthermore, with only 100 labeled examples, it matches the performance of training from scratch on 100× more data. We open-source our pretrained models and code[1].

While Deep Learning models have achieved state-of-the-art on many NLP tasks, these models are trained from scratch, requiring large datasets, and days to converge. Research in NLP focused mostly on *transductive* transfer (Blitzer et al., 2007). For *inductive* transfer, fine-tuning pre-trained word embeddings (Mikolov et al., 2013), a simple transfer technique that only targets a model's first layer, has had a large impact in practice and is used in most state-of-the-art models. Recent approaches that concatenate embeddings derived from other tasks with the input at different layers (Peters et al., 2017; McCann et al., 2017; Peters et al., 2018) still train the main task model from scratch and treat pretrained embeddings as fixed parameters, limiting their usefulness.

In light of the benefits of pretraining (Erhan et al., 2010), we should be able to do better than *randomly initializing* the remaining parameters of our models. However, inductive transfer via fine-tuning has been unsuccessful for NLP (Mou et al., 2016). Dai and Le (2015) first proposed fine-tuning a language model (LM) but require millions of in-domain documents to achieve good performance, which severely limits its applicability.

We show that not the idea of LM fine-tuning but our lack of knowledge of how to train them effectively has been hindering wider adoption. LMs

1 Introduction

Inductive transfer learning has had a large impact on computer vision (CV). Applied CV models (including object detection, classification, and segmentation) are rarely trained from scratch, but instead are fine-tuned from models that have been pretrained on ImageNet, MS-COCO, and other datasets (Sharif Razavian et al., 2014; Long et al.,

Question

1. Does TL Possible in NLP?
2. ... FT ... ?
3. How ?

Language modelling ⇒

1. classification
2. generation
3. Summarization
4. translation

I am sunny lovely

I am sunny	am sunny lovely

lovely ?

translation

I am sunny | मैं धूप में हूँ

training

Data → Model → Prediction

Supervised Learning

Supervised language modelling

(teaching to my model)

Descriminative modelling — Less data, Specific task

Creative modelling x — More data, general task

LLM → LLM

Unsupervised Language modelling

Huge data → Label not required

Task ⇒ Data generation (Next word Prediction)

text

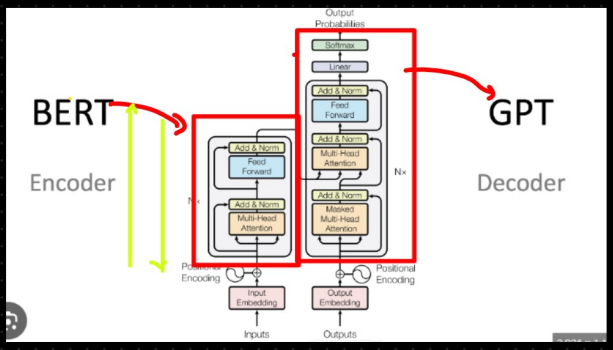
How ::
1. Statical Pattern (Dist.)
2. Semantic word (word-word)
3. Contextual relationship

Supervised lang. Modelling

Label will be req

(1) So that Model can learn
(2) based on Previous word can Predict next word

10 BERT



Language Model

↓

huge amount

⇒ ULM

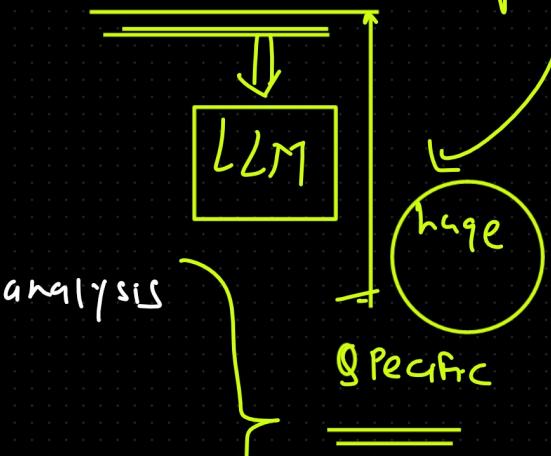
↓

Google

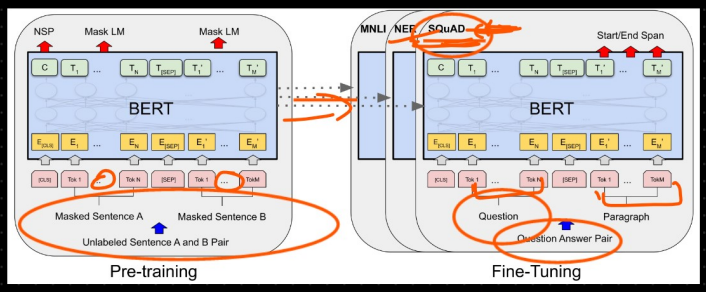
teching

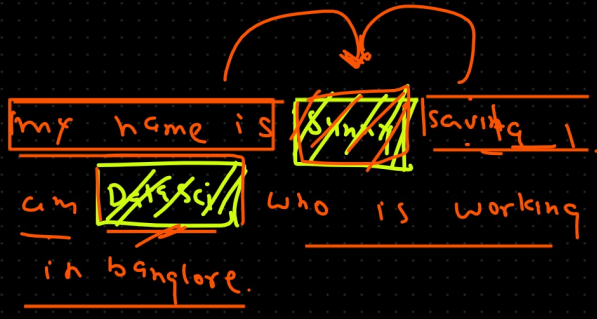
(next word Pre)

1. Encoder only model
2. MLM
3. NWP, NSP
4. Language Model
   1. Unsupervised Pretraining
   2. Supervised Fine tuning
5. Bi-Directional Model ⇒



6. Use-Case ⇒
   1. Sentiment analysis
   2. NER
   3. Pos
   4. question Answering





11 GPT ⇒ Generative Pre training

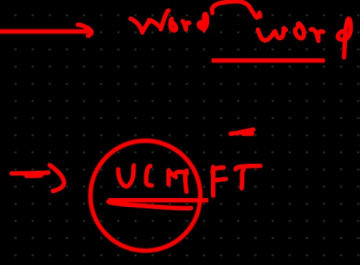
GPT1 :Improving Language Understanding by Generative Pre-Training

Generative Pre-training:

The term "generative" refers to the model's capability to generate coherent and contextually relevant text. "Pre-training" indicates that the model is initially trained on a large corpus of text data in an unsupervised manner before being fine-tuned for specific tasks.
During the pretraining phase, the model learns to predict the next word in a sequence, capturing language patterns and semantics.

Key Objectives:

The paper outlines the methodology of training a transformer-based language model on a diverse range of tasks without task-specific labelled data.
The generative pretraining approach allows the model to acquire a broad understanding of language, enabling it to perform well on various downstream tasks.



GPT2: language models are unsupervised multitask leaner

In the context of the GPT-2 (Generative Pre-trained Transformer 2) paper, the term "unsupervised multitask learner" refers to the model's ability to perform a variety of natural language processing (NLP) tasks without task-specific supervision during the pretraining phase. GPT-2 is designed as a large-scale language model that is pretrained on a diverse corpus of text data without explicit annotations for specific tasks. given to it at runtime

⇒ VLM → huge

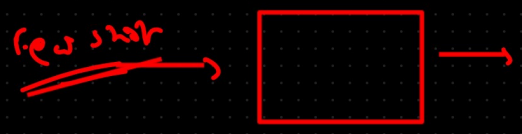
GPT3: language model are fewshot learner

In the context of the GPT-3 research paper, the term "few-shot learner" refers to the model's ability to perform a task or answer questions with only a few examples or prompts provided during inference. GPT-3 is known for its remarkable ability to generalise from a small number of examples or demonstrations given to it at runtime.

⇒ very huge

175B

Aspect	GPT-1	GPT-2	GPT-3
Model Size	117 million parameters	1.5 billion parameters	Up to 175 billion parameters
Context Window Size	Limited due to model size	Larger context window	Extremely large context window
Fine-Tuning Capabilities	Limited fine-tuning capabilities	Improved fine-tuning capabilities	Enhanced fine-tuning capabilities
Released Date	June 2018	February 2019	June 2020
Prompt Engineering	Limited influence on model behavior	More influence on model behavior	Substantial influence on model behavior
Downstream Task Performance	Less impressive on specific tasks	Improved performance on diverse tasks	Exceptional performance on wide range
Ethical Considerations	Earlier models raised ethical concerns	Increased awareness and scrutiny	Continued focus on addressing biases
Research Impact	Pioneered large-scale unsupervised learning	Advanced understanding of transfer learning	Broader applications and benchmarks
Use Cases and Applications	Limited by model size and capabilities	Broader applications in various domains	Extensive applications across industries
Public Access	Limited access initially	Increased access with GPT-2	OpenAI API access for GPT-3



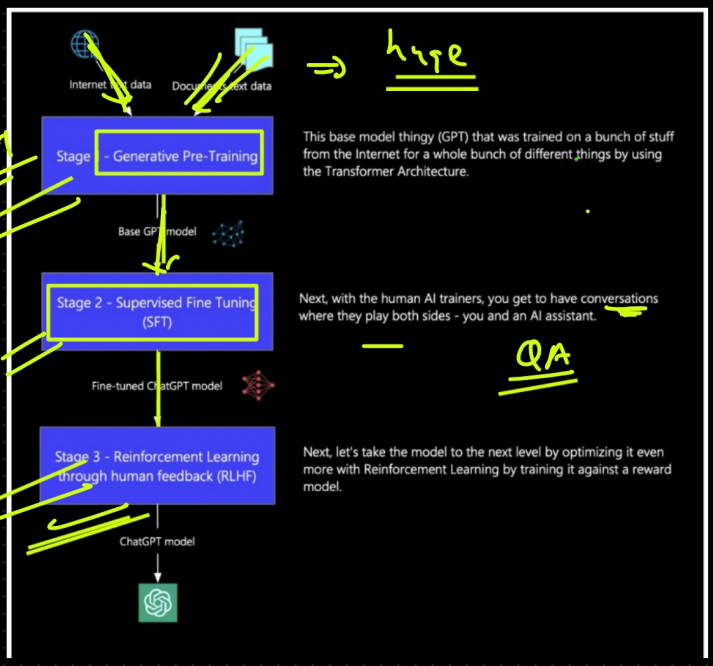
12 Training of chatgpt ⇒ Application

↑

GPT-3

↓

3.5 | 3.5 turbo , gpt-4, 4turbo



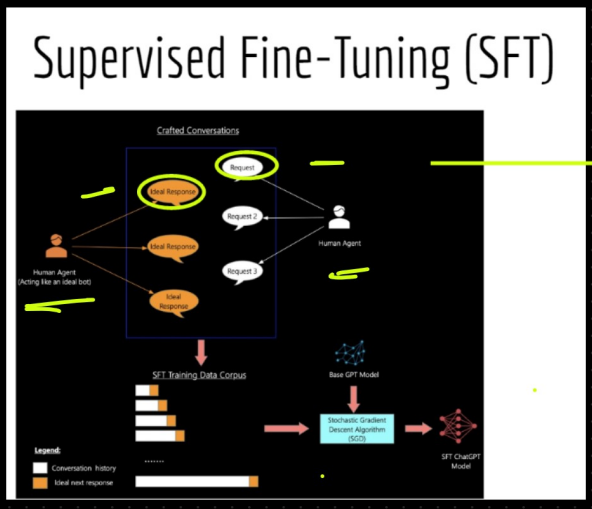
ULM

⇒ hype

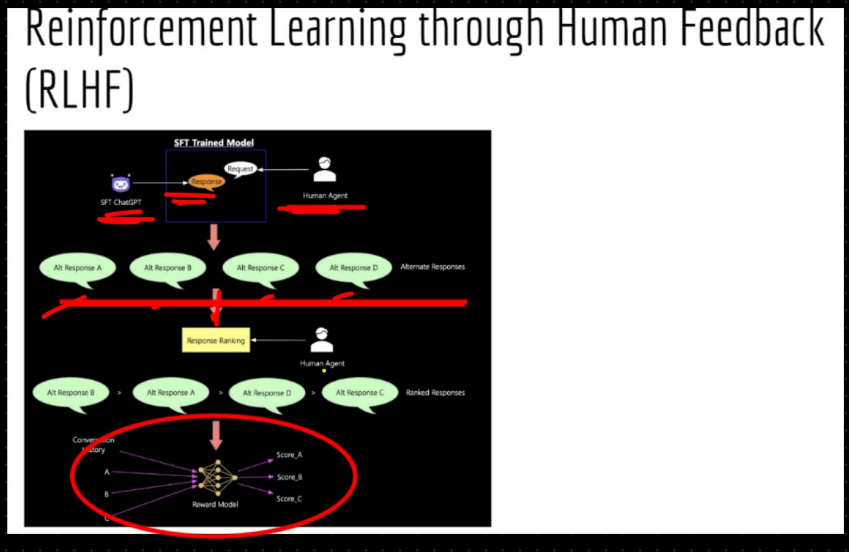
QA

→ transformer

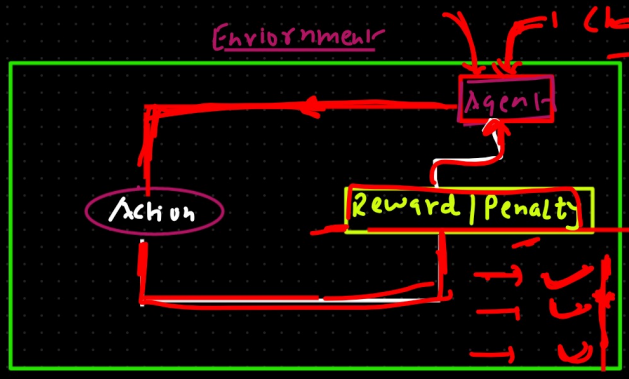
; dIA



9 4 6 r q
stackoverflow
r-b i hsfq
Reddit



RLHF



⇒ Q^*, Q Learning

⇒ Deep Q learning

Imp Points

1. Ethics
2. Maintaing Memory content
3. Dialogue specific training ⇒ Conversation
4. RHFL (iterative Feedback)
5. Bias / error

Conclusions →