Welcome to SEAS Online at George Washington University

Class will begin shortly

Audio: To eliminate background noise, please be sure your audio is muted. To speak, please click the hand icon at the bottom of your screen (**Raise Hand**). When instructor calls on you, click microphone icon to unmute. When you've finished speaking, ***be sure to mute yourself again***.

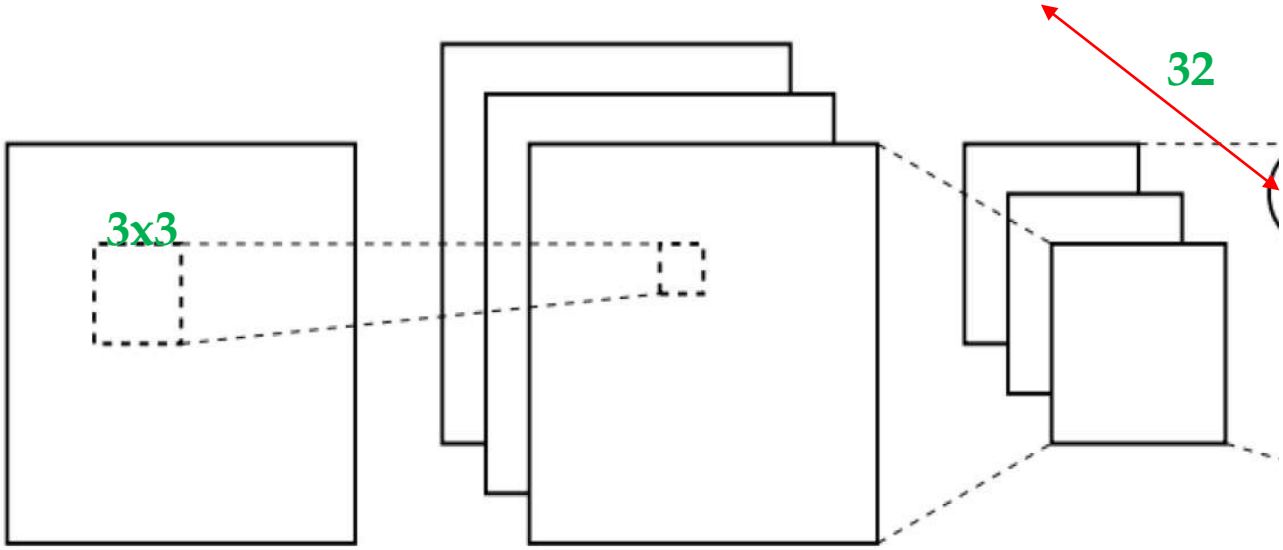
Chat: Please type your questions in Chat.

Recordings: As part of the educational support for students, we provide downloadable recordings of each class session to be used exclusively by registered students in that particular class. Releasing these recordings is strictly prohibited.

Shearing

Src:https://datamonje.com/image-data-augmentation/

Convolution Matrix Size Calculation



28x28

26x26 x 32

S=1, p=0 f=3

$\frac{26-2}{2} + 1 = 13$ (13x13)

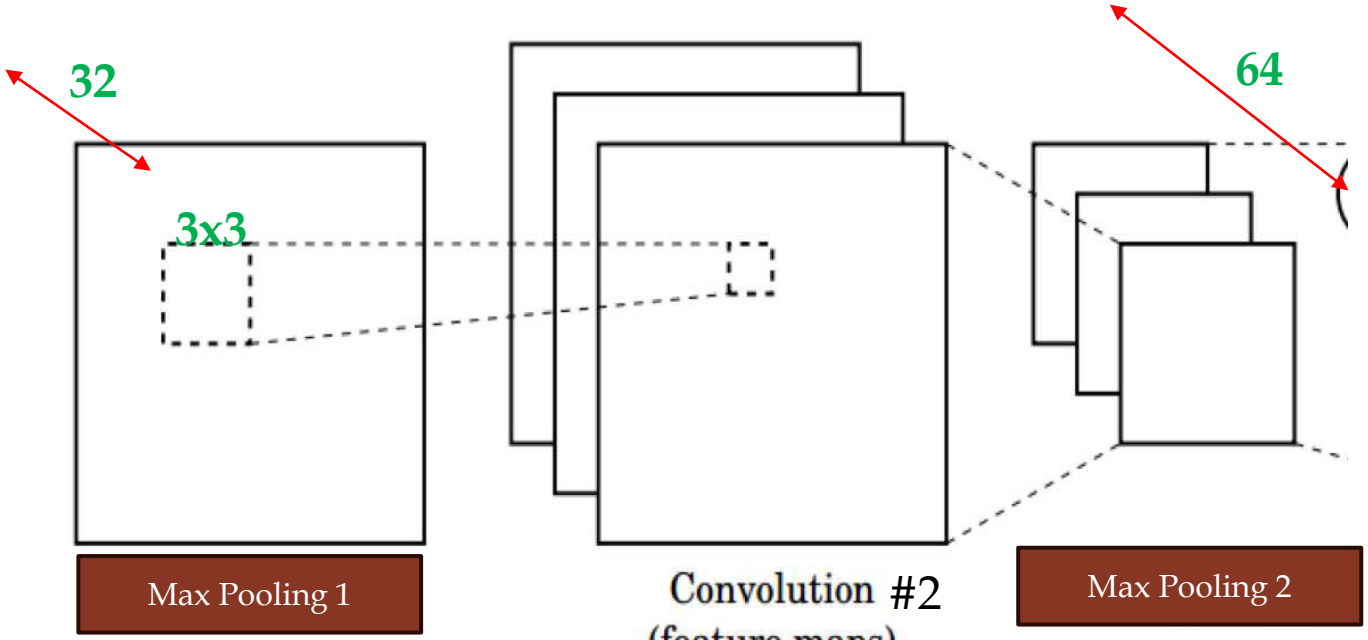
S=2,p=0,f=2

$$\left(\left\lfloor \frac{n+2p-f}{s} + 1 \right\rfloor\right) \times \left(\left\lfloor \frac{n+2p-f}{s} + 1 \right\rfloor\right)$$

$$Output: \left\lfloor \frac{n_H - f}{s} + 1 \right\rfloor \times \left\lfloor \frac{n_W - f}{s} + 1 \right\rfloor \times n_c$$

Floor operation: round down

Convolution Matrix Size Calculation



13x13

11x11

S=1, p=0 f=3

$\frac{11-2}{2} + 1 = 5$ (5x5)

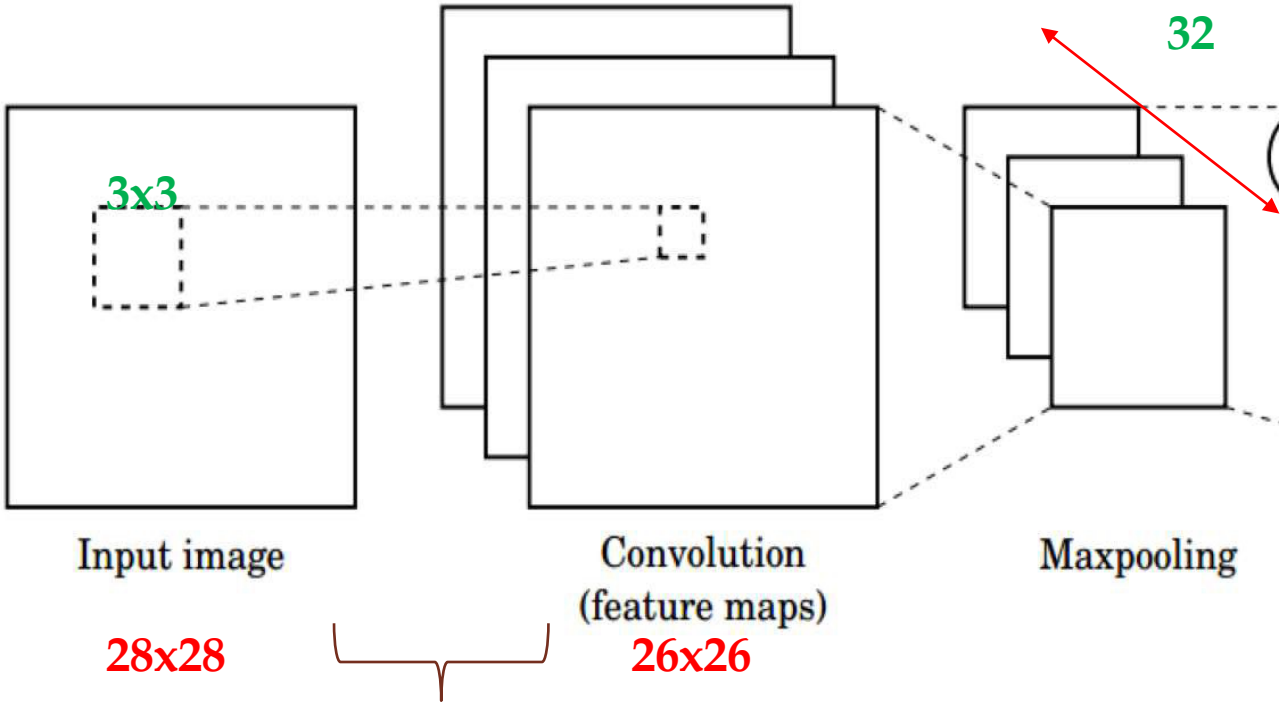
S=2,p=0,f=2

$$\left(\left\lfloor \frac{n+2p-f}{s} + 1 \right\rfloor\right) \times \left(\left\lfloor \frac{n+2p-f}{s} + 1 \right\rfloor\right)$$

$$Output: \left\lfloor \frac{n_H - f}{s} + 1 \right\rfloor \times \left\lfloor \frac{n_W - f}{s} + 1 \right\rfloor \times n_c$$

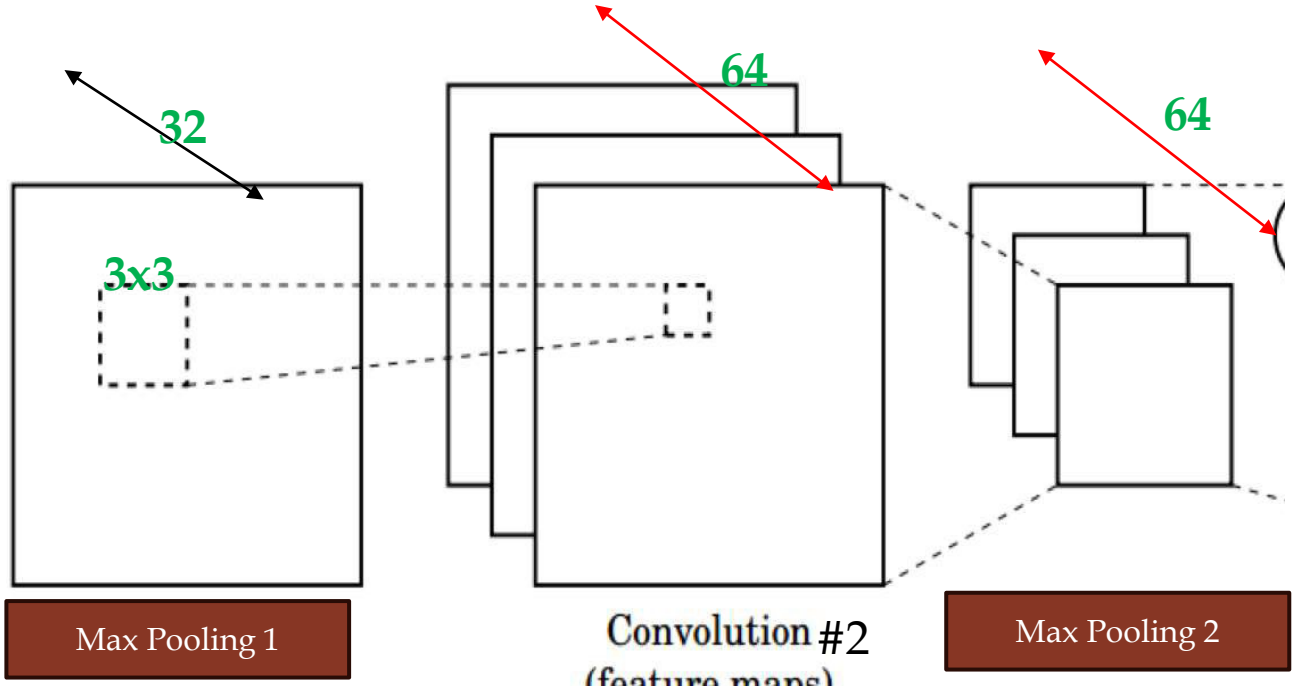
Floor operation: round down

Convolution Params Calculation



Learnable Params:
3x3 = 9 filter size.
32 Filters = 9 * 32
32 Biases
TOTAL = 320

Convolution Params Calculation



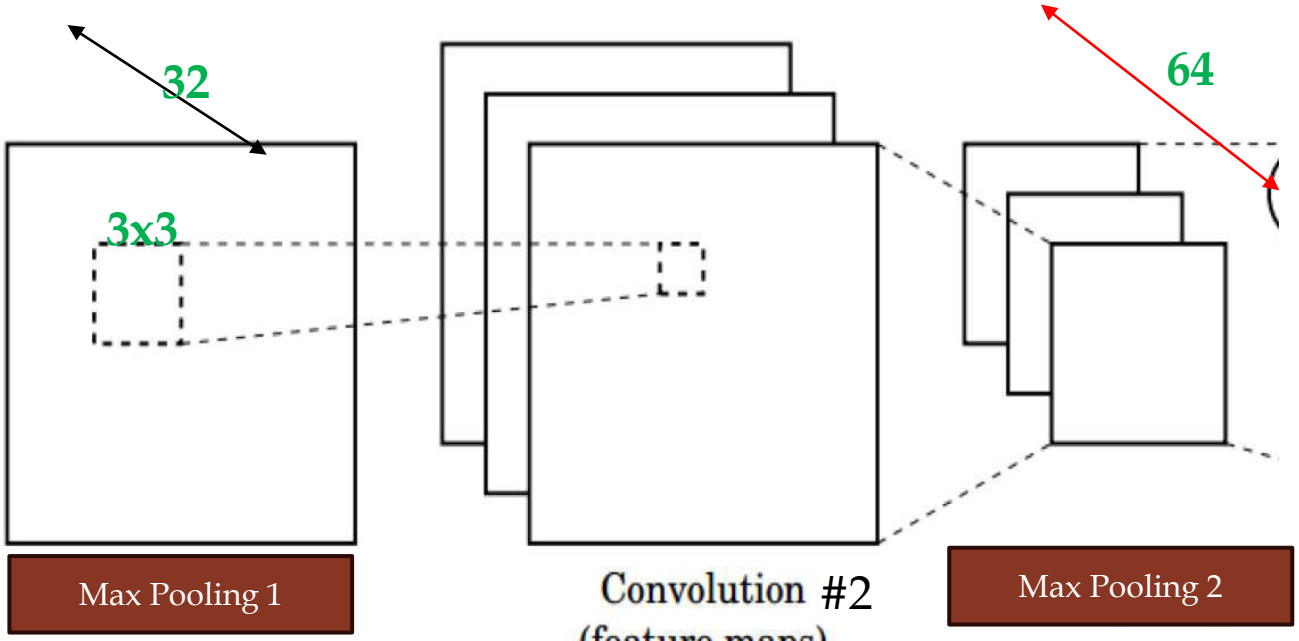
Learnable Params:
3x3 = 9 filter size/plane.
Filter Depth = 32
64 Filters.

Total = 9x32x64=

+ 64 Biases
TOTAL = 18,496

Total Parameters=(Filter Height×Filter Width×Number of Input Channels+1)×Number of Filters
Total Parameters = (3x3x32 + 1) * 64 = 18496

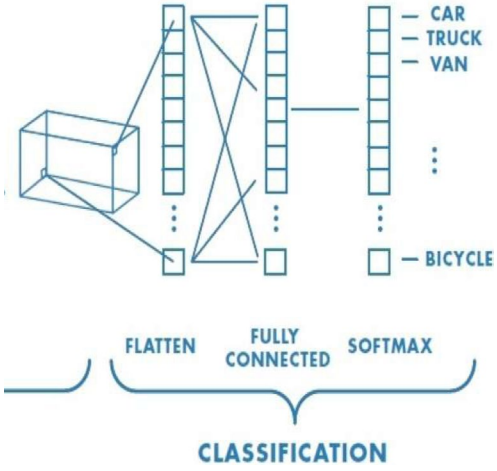
Convolution Params Calculation



Total Parameters=(Filter Height×Filter Width×Number of Input Channels+1)×Number of Filters
Total Parameters = (3x3x32 + 1) * 64 = 18496

Convolution Params Calculation

5x5x64

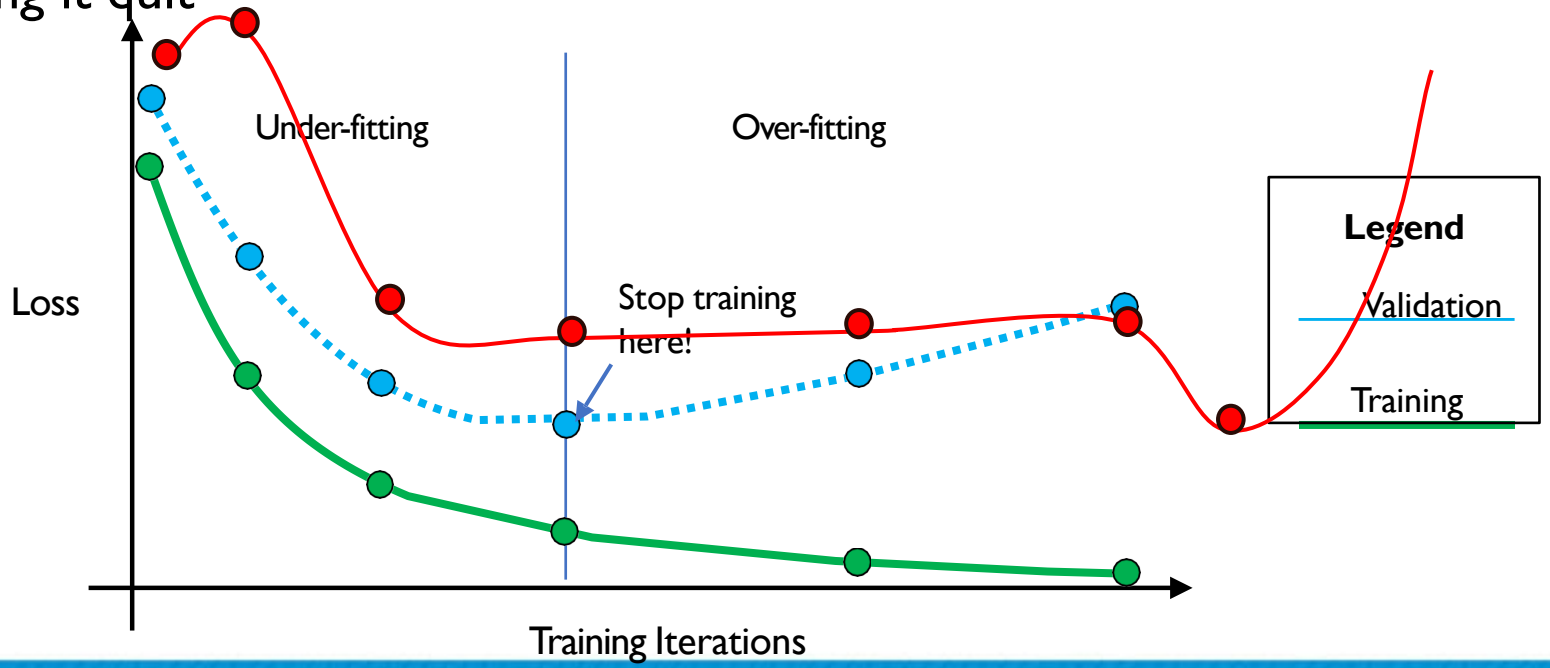


Flattening the pooled layer is 5x5x64 = 1600 inputs … x 128 neurons in dense layer = 204,800
We have also the 128 biases = 204800 + 128 = 204,928

Last Layer has 10 neurons = 10 * 128 + 10 = 1290 Parameters.

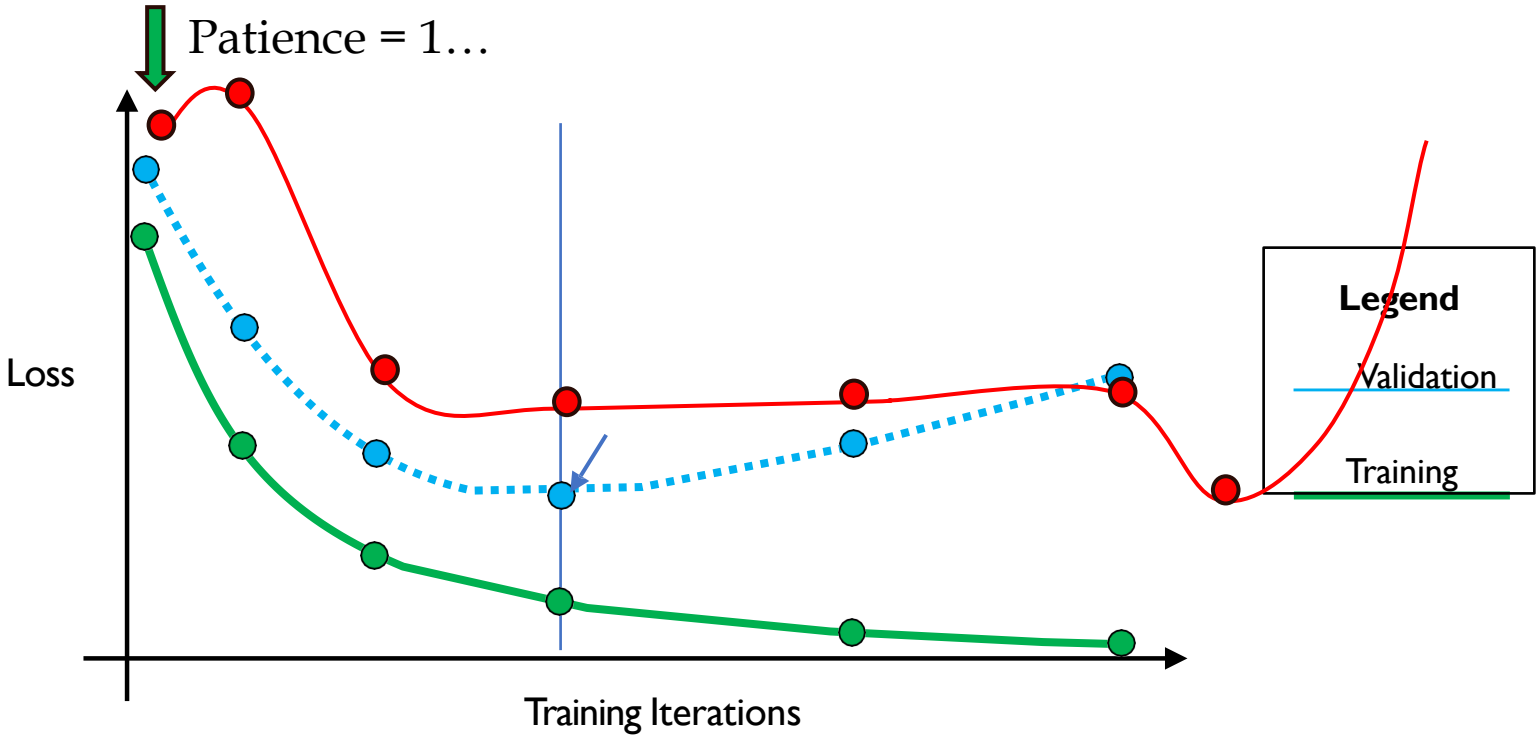
Regularization 2: Early Stopping

- Min_delta: What is the least change in loss that I would consider as "acceptable" to change my baseline loss. If it is too small (0.0000001), it Is not worth it to count as improvement (and if the loss is more, even worse news)
- Patience: How much "trials"="epochs" to wait on improvement before calling it quit



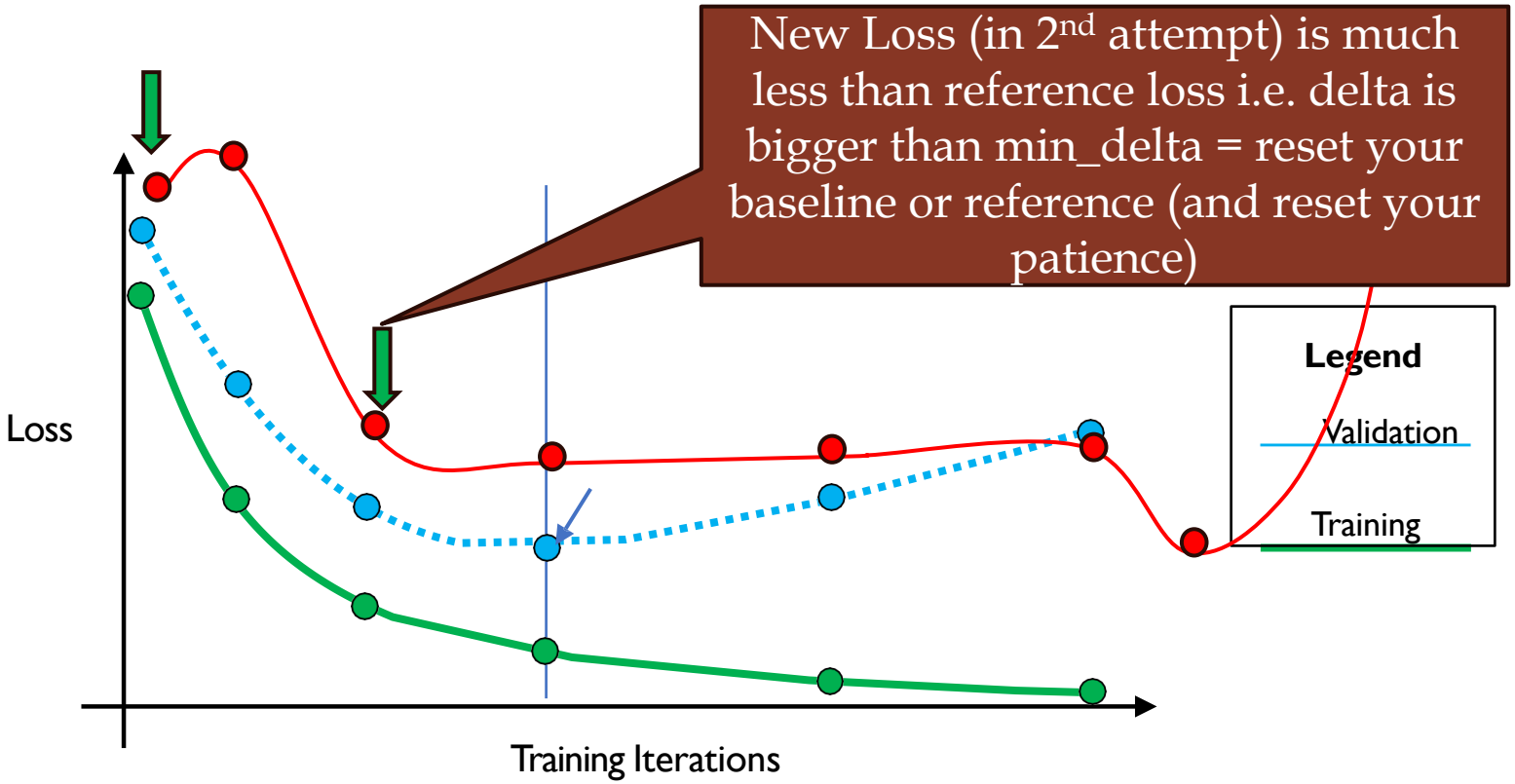
Source: introtodeeplearning.com

Regularization 2: Early Stopping



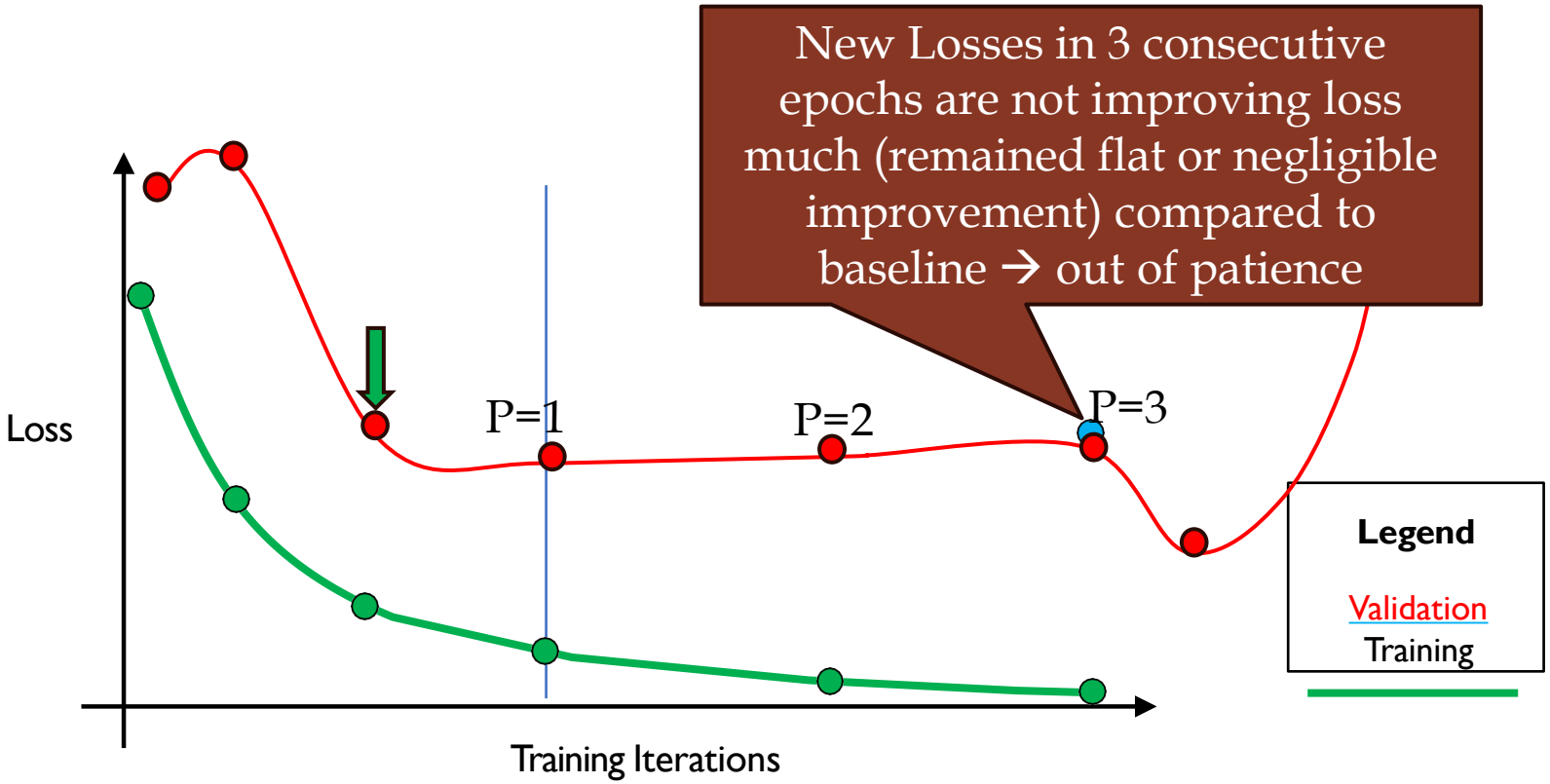
Source: introtodeeplearning.com

Regularization 2: Early Stopping



Source: introtodeeplearning.com

Regularization 2: Early Stopping



Source: introtodeeplearning.com

THE GEORGE WASHINGTON UNIVERSITY
WASHINGTON, DC

SEAS 8520 – Lecture 6: Intro to NLP

Walid Hassan, M.B.A, D.Eng.

AI & Ethics

“ This potent technology will change the world to at least the same extent as electricity, the internal combustion engine, the transistor, or the internet. The potential benefits in healthcare, design, entertainment, transport, education, and almost every area of commerce are enormous. However, scientists and engineers are often unrealistically optimistic about the outcomes of their work, and the potential for harm is just as great.”

Src: Prince, S. J. (January 28, 2024). Understanding Deep Learning. MIT Press. http://udlbook.com

AI & Ethics

Bias and Fairness: When AI systems are trained on historical data, they may inherently reproduce existing biases present in that data. For instance, a model predicting salary levels might reflect historical gender wage gaps, suggesting lower salaries for women compared to men. This is not a theoretical concern but one that has manifested in various AI applications, drawing international attention. Examples include an AI system that altered the facial features of non-white individuals to appear more white in super-resolved images, and another that generated images exclusively of men when tasked with creating pictures of lawyers. The careless deployment of AI in decision-making can, therefore, reinforce or exacerbate societal biases.

This issue underscores the critical need for vigilance and ethical considerations in AI development and application. It highlights the potential of AI systems to perpetuate and institutionalize bias, making it imperative to integrate fairness and bias mitigation strategies in AI models.

Src: Prince, S. J. (January 28, 2024). Understanding Deep Learning. MIT Press. http://udlbook.com

AI & Ethics

Explainability: Deep learning models, characterized by their complex architectures and billions of parameters, often operate as "black boxes." While these systems can make decisions or predictions, understanding the specifics of how these decisions are derived remains a challenge. This lack of transparency has spurred the development of explainable AI (XAI), a sub-field focused on making AI decision-making processes comprehensible.

Although fully elucidating the workings of deep learning systems may be currently out of reach, efforts have been made to provide local explanations. These aim to offer interpretable insights into why a particular decision was made by an AI model, without necessarily unpacking the model's overall logic. However, the feasibility of creating complex, yet fully transparent decision-making systems remains an open question.

Src: Prince, S. J. (January 28, 2024). Understanding Deep Learning. MIT Press. http://udlbook.com

AI & Ethics

Weaponizing AI : Historically, major technological advancements have found applications in warfare, reflecting the unfortunate persistence of violent conflict in human society. Given its profound capabilities, AI is no exception and is likely to be, and in some cases already is, utilized extensively in military contexts. This raises ethical, moral, and existential concerns about the applications of AI technology.

The potential for AI to be weaponized necessitates a global dialogue on ethics, regulations, and controls to prevent misuse and ensure that the development and deployment of AI technologies align with humanitarian values and principles.

AI & Ethics

Concentrating Power: The pursuit of artificial intelligence by the world's most powerful companies is driven not by a philanthropic desire to better humanity but by the promise of substantial profits. Deep learning and other advanced technologies have the potential to centralize power within a select few organizations that master them. One significant concern is the automation of jobs, which threatens to disrupt the economic landscape and adversely impact lower-paid, less-skilled workers. While some optimists draw parallels to the industrial revolution, which ultimately led to reduced working hours, the truth remains uncertain regarding AI's comprehensive impact on society.

This scenario invites a critical examination of how technological advancements, particularly AI, can reshape societal structures, economic models, and the distribution of power. It prompts students to reflect on historical technological disruptions and consider ethical, equitable frameworks for AI development and deployment. Moreover, it raises questions about safeguarding against exacerbating social inequalities and finding pathways to inclusive growth.

Src: Prince, S. J. (January 28, 2024). Understanding Deep Learning. MIT Press. http://udlbook.com

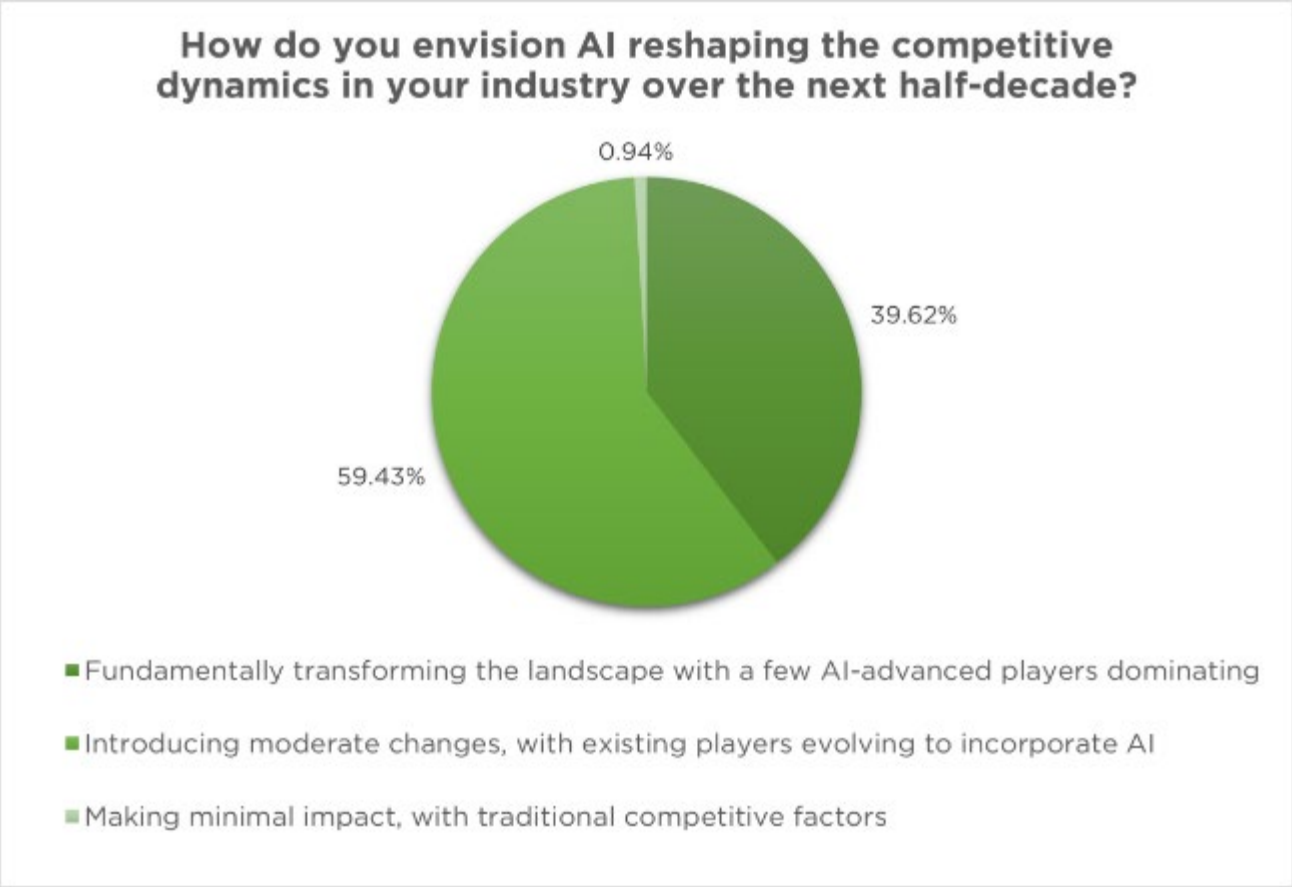
AI & Ethics

Existential Risk: Historically, the most significant existential threats to humanity have emerged from technological advancements. Climate change, nuclear warfare, and pandemics highlight how innovations, despite their benefits, can precipitate global crises. Artificial intelligence introduces unprecedented existential risks, necessitating extreme caution in developing systems potentially surpassing human capabilities. Optimistically, AI could concentrate immense power in the hands of its controllers. Pessimistically, it might become uncontrollable or inscrutable, posing threats we cannot yet fully comprehend.

The existential risks associated with AI underscore the need for cautious, ethical engagement with this technology. Students are encouraged to debate the balance between innovation and control, considering how safeguards, ethical guidelines, and international cooperation might mitigate these risks. This topic also invites exploration of speculative scenarios, both optimistic and pessimistic, encouraging students to engage with a wide range of viewpoints and potential outcomes.

Src: Prince, S. J. (January 28, 2024). Understanding Deep Learning. MIT Press. http://udlbook.com

Korn Ferry report on AI



Src: https://www.kornferry.com/institute/humans-still-wanted-the-future-of-work-in-an-ai-driven-world?utm_source=marketo&utm_medium=em&utm_content=article&utm_campaign=23-10-10-csuite-newsletter&mkt_tok=MjUxLU9MUi05NTgAAAGOuXUmtd8Ofrzmo11KqKIrEO0O9IwW_J2qr6V71AfhAylaCSjg45XvQ_Ht3FYO_6NyfCoGW2mxQF0Gz7Q5_-rLvnarRBmSghQ-aI0LfNYy524

Korn Ferry report on AI

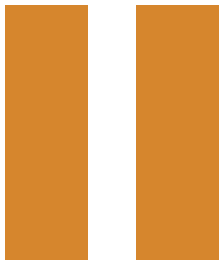
- 82% of CEOs and senior leaders we surveyed believe AI will have an extreme to significant impact on their business
- CEOs and senior executives planning to spend up to 50% more on AI-related strategies
- more than 33% of senior leaders surveyed say they are already experimenting with ways to leverage AI to boost productivity and operating efficiency
- When asked to identify the biggest obstacle to AI integration, 40% of respondents cited a lack of AI-related knowledge and skills within their HR team, while roughly 13% named employee resistance to change management issues
- The rise of AI in business is imminent and unstoppable. After all, AI promises immense potential—for business growth, for efficiency, and for innovation. But it's essential to remember that AI and humans are not competitors in this new world

Src: https://www.kornferry.com/institute/humans-still-wanted-the-future-of-work-in-an-ai-driven-world?utm_source=marketo&utm_medium=em&utm_content=article&utm_campaign=23-10-10-csuite-newsletter&mkt_tok=MjUxLU9MUi05NTgAAAGOuXUmtd8Ofrzmo11KqKlrEO0O9IwW_J2qr6V71AfhAylaCSjg45XvQ_Ht3FYO_6NyfCoGW2mxQF0Gz7Q5_-rLvnarRBmSghQ-aI0LfNYy524

WSJ – World Economic Forum

- Nadella said he expected many positive impacts for workers, arguing they would become more productive and should see wages rise accordingly.
- The world had a two-week freakout with GPT4," "And now, people are like, 'Why is it so slow?'" Altman said.
- "We always find new things to do, and yet it does seem somewhat different if AI can, like, have more cognitive power than any of us," he said. "We have no idea what happens next."

https://www.wsj.com/tech/ai/altman-and-nadella-talk-ai-at-davos-18ed2fe0?mod=ai_more_article_pos3



BREAK

Please come back @

2:34 PM EST

1:34 PM CST

What is NLP

- NLP is a branch of artificial intelligence that deals with analyzing, understanding, and generating the languages that humans use naturally in order to interface with computers in both written and spoken contexts using natural human languages instead of computer languages

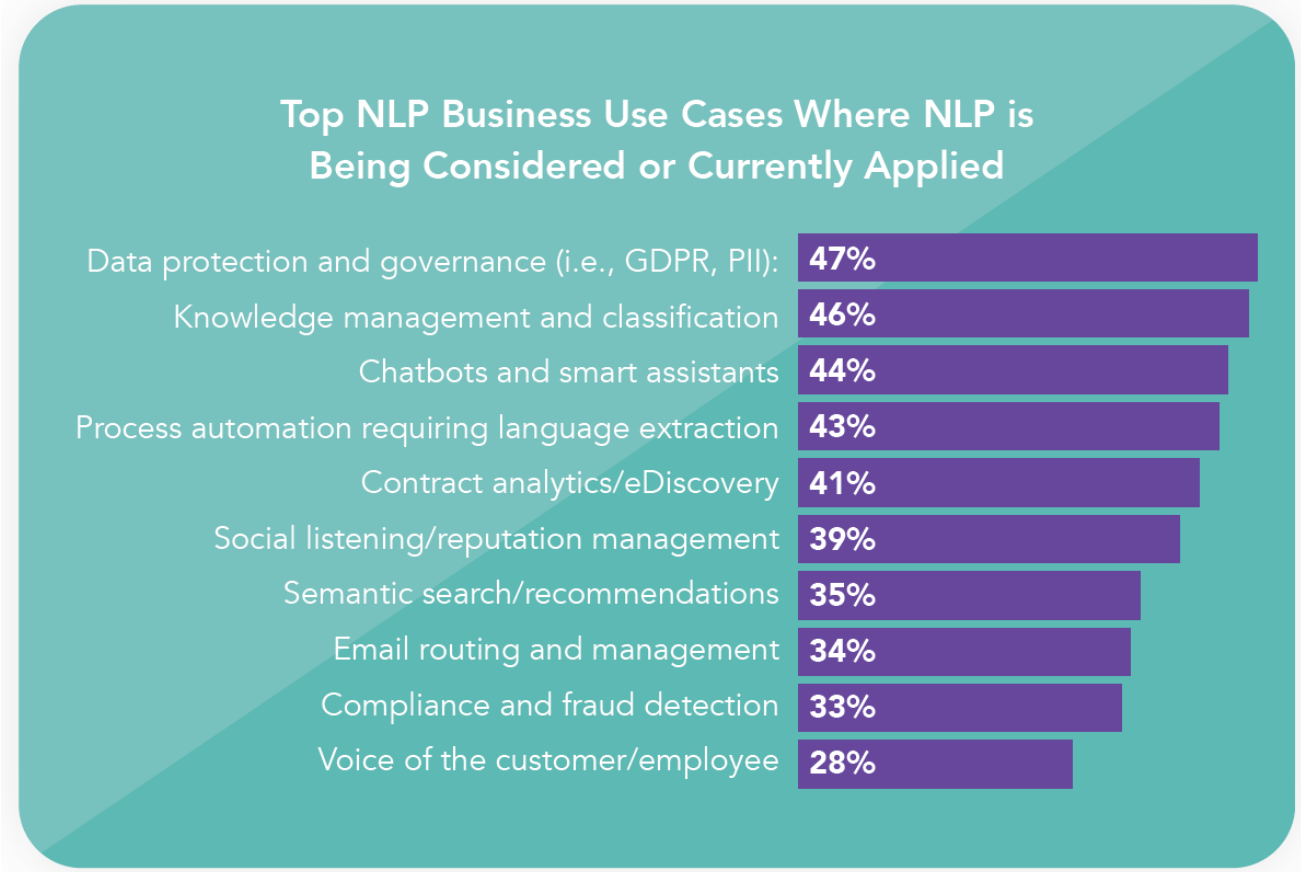
Applications:

- Machine translation(Google Translate)
- Natural language generation
- Web Search
- Spam filters
- Sentiment Analysis
- Chatbots… and many more

Src:Gareth James, D. W. (2023). An Introduction to Statistical Learning: with Applications in Python. Springer.

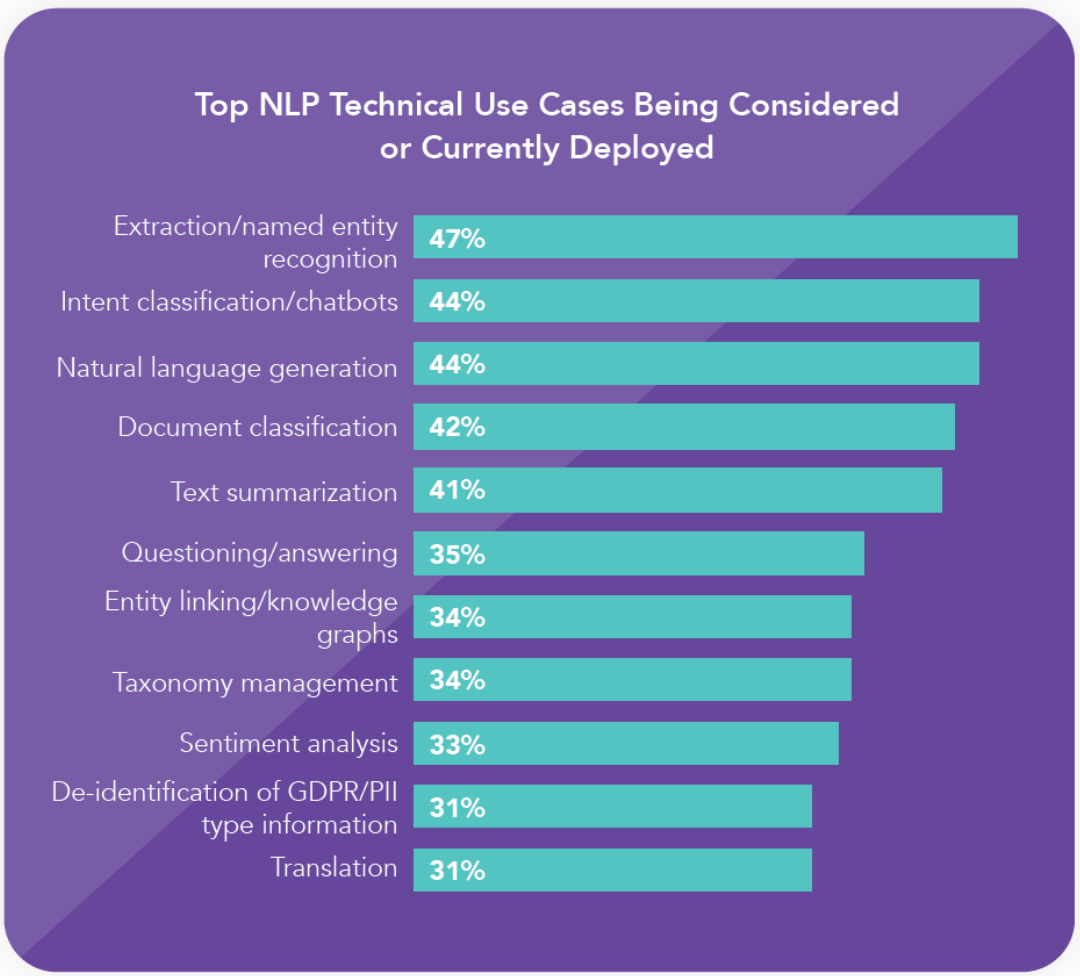
https://towardsdatascience.com/a-gentle-introduction-to-natural-language-processing-e716ed3c0863

Top NLP Use Cases



Src:https://www.expert.ai/wp-content/uploads/2022/12/The-2023-Expert-NLP-Survey-Report-Trends-driving-NLP-Investment-and-Innovation.pdf

Top NLP Use Cases



Src:https://www.expert.ai/wp-content/uploads/2022/12/The-2023-Expert-NLP-Survey-Report-Trends-driving-NLP-Investment-and-Innovation.pdf

NLP Challenges by Maturity Level

Top 3 Challenges by NLP Maturity Level		
Evaluation and experimentation phase (no NLP models in production)	**Early phase NLP implementation (under two years in production)**	**More mature NLP models (two years-plus in production)**
Data security and governance	Aligning with business stakeholders on which NLP uses cases to prioritizegovernance	Building the business case/ ROI for NLP projects
Building the business case/ ROI for NLP projects	Data security and governance	Achieving the level of accuracy/quality needed to put NLP models in production
Choosing which artificial intelligence approaches to use to achieve the desired model result	Costs associated with NLP modeling and tools	Costs associated with NLP modeling and tools

Src:https://www.expert.ai/wp-content/uploads/2022/12/The-2023-Expert-NLP-Survey-Report-Trends-driving-NLP-Investment-and-Innovation.pdf

NLP & Cybersecurity

Sample key applications of NLP in Cybersecurity

Threat Intelligence:

- **Automated Analysis**: Processing and analyzing vast amounts of textual data from sources such as blogs, forums, and news articles to identify new threats.
- **Entity Recognition**: Extracting relevant cybersecurity entities like threat actor names, malware names, or target organizations from textual data.
- **Topic Modeling**: Understanding dominant themes or topics from a large collection of cyber threat intelligence reports.

Src:Gareth James, D. W. (2023). An Introduction to Statistical Learning: with Applications in Python. Springer.

https://towardsdatascience.com/a-gentle-introduction-to-natural-language-processing-e716ed3c0863

NLP & Cybersecurity

Phishing Detection:

- **Email Analysis**: Analyzing email content to determine if it's a phishing attempt based on the text, structure, or sentiment.
- **URL Analysis:** Analyzing URLs present in emails or messages to check for malicious intent or patterns.

Security Awareness and Training:

- **Chatbots**: Implementing NLP-driven chatbots for security awareness training and for answering common cybersecurity queries.
- **Sentiment Analysis**: Assessing employee feedback on security training modules to optimize content.

Src:Gareth James, D. W. (2023). An Introduction to Statistical Learning: with Applications in Python. Springer.

https://towardsdatascience.com/a-gentle-introduction-to-natural-language-processing-e716ed3c0863

NLP & Cybersecurity

Log Analysis:

- **Anomaly Detection**: Processing and analyzing logs with NLP to detect anomalies or suspicious textual patterns.
- **Semantic Search**: Searching through logs using natural language queries.

Incident Reports and Management:

- **Automated Triage**: Analyzing incident reports to automatically classify and prioritize them based on content.
- **Trend Analysis**: Identifying recurring themes or patterns in incident reports to predict potential future threats.

Src:Gareth James, D. W. (2023). An Introduction to Statistical Learning: with Applications in Python. Springer.

https://towardsdatascience.com/a-gentle-introduction-to-natural-language-processing-e716ed3c0863

NLP & Cybersecurity

Social Media Monitoring:

- Monitoring social media platforms for potential security threats, data leaks, or discussions about vulnerabilities related to an organization.

Automated Response:

- Chatbots and Virtual Assistants: Using NLP-driven solutions to guide users during security incidents or when security actions are needed.

Data Loss Prevention (DLP):

- **Content Analysis**: Analyzing the content of documents to ensure sensitive information is not being shared or leaked outside of permitted channels.

And Many more….

Src:Gareth James, D. W. (2023). An Introduction to Statistical Learning: with Applications in Python. Springer.

https://towardsdatascience.com/a-gentle-introduction-to-natural-language-processing-e716ed3c0863

History of NLP

Rule-based machine translation (~1970):

- The main approach of RBMT systems is based on linking the structure of the given input sentence with the structure of the demanded output sentence, necessarily preserving their unique meaning.
- The following example can illustrate the general frame of RBMT:

  A girl eats an apple. Source Language = English; Demanded Target Language = German

  Minimally, to get a German translation of this English sentence one needs:

  A dictionary that will map each English word to an appropriate German word.

  Rules representing regular English sentence structure.

  Rules representing regular German sentence structure.

Src:Gareth James, D. W. (2023). An Introduction to Statistical Learning: with Applications in Python. Springer.
https://en.wikipedia.org/wiki/Rule-based_machine_translation

History of NLP

- The Late 1980s: The realization that purely rule-based systems had significant limitations started gaining traction. The availability of digital texts and increased computational power made statistical methods feasible.
- 1990s: This decade saw a surge in the adoption of statistical methods in NLP. Techniques like statistical machine translation, probabilistic context-free grammars (PCFGs), and others became popular.
- **Brown Corpus**: While created in the 1960s, the significance of the Brown Corpus (and other corpora that followed) can't be understated. Having a large dataset of English text allowed researchers to experiment with data-driven techniques in linguistics and NLP.

Src:Gareth James, D. W. (2023). An Introduction to Statistical Learning: with Applications in Python. Springer.
https://en.wikipedia.org/wiki/Rule-based_machine_translation

History of NLP

- IBM's Candide: In the early 1990s, IBM introduced "Candide," one of the first data-driven machine translation systems based on statistical models. This marked a significant departure from the rule-based machine translation methods that had been dominant.
- From the 1990s onward, statistical models became the dominant approach in NLP until the rise of deep learning methods in the 2010s, which integrated neural networks with statistical techniques.

Src:Gareth James, D. W. (2023). An Introduction to Statistical Learning: with Applications in Python. Springer.
https://en.wikipedia.org/wiki/Rule-based_machine_translation

History of NLP

- Statistical Method involve using the “pre-ANNs” techniques.
- For Example, assume we want to do a sentiment analysis for our text. we can use Term Frequency – Inverse Document Frequency (TF-IDF) (e.g. scikit-learn) to first compute the frequency of words, and use Random Forests, SVMs to train a model based on a certain dataset.

Src:Gareth James, D. W. (2023). An Introduction to Statistical Learning: with Applications in Python. Springer.
https://en.wikipedia.org/wiki/Rule-based_machine_translation

History of NLP

- TF-IDF: IDF =Log[(# Number of documents) / (Number of documents containing the word)] and TF = (Number of repetitions of word in a document) / (# of words in a document)

Document 1 It is going to rain today.

Document 2 Today I am not going outside.

Document 3 I am going to watch the season premiere.

https://medium.com/analytics-vidhya/tf-idf-term-frequency-technique-easiest-explanation-for-text-classification-in-nlp-with-code-8ca3912e58c3

History of NLP

Document 1 It is going to rain today. TF = (Number of repetitions of word in a document) / (# of words in a document)

Words/ Documents	Document 1
going	0.16
to	0.16
today	0.16
i	0
am	0
it	0.16
is	0.16
rain	0.16

TF for sentence 1

https://medium.com/analytics-vidhya/tf-idf-term-frequency-technique-easiest-explanation-for-text-classification-in-nlp-with-code-8ca3912e58c3

History of NLP

Words/ Documents	Document 1	Document 2	Document 3
going	0.16	0.16	0.12
to	0.16	0	0.12
today	0.16	0.16	0
i	0	0.16	0.12
am	0	0.16	0.12
it	0.16	0	0
is	0.16	0	0
rain	0.16	0	0

TF for the document

https://medium.com/analytics-vidhya/tf-idf-term-frequency-technique-easiest-explanation-for-text-classification-in-nlp-with-code-8ca3912e58c3

History of NLP

Words	IDF Value
going	log(3/3)
to	log(3/2)
today	log(3/2)
i	log(3/2)
am	log(3/2)
It	log(3/1)
is	log(3/1)
rain	log(3/1)

https://medium.com/analytics-vidhya/tf-idf-term-frequency-technique-easiest-explanation-for-text-classification-in-nlp-with-code-8ca3912e58c3

History of NLP

Words	IDF Value
going	0
to	0.41
today	0.41
i	0.41
am	0.41
It	1.09
is	1.09
rain	1.09

Words/ Documents	Document 1	Document 2	Document 3
going	0.16	0.16	0.12
to	0.16	0	0.12
today	0.16	0.16	0
i	0	0.16	0.12
am	0	0.16	0.12
it	0.16	0	0
is	0.16	0	0
rain	0.16	0	0

IDF Value and TF value of 3 documents.

https://medium.com/analytics-vidhya/tf-idf-term-frequency-technique-easiest-explanation-for-text-classification-in-nlp-with-code-8ca3912e58c3

History of NLP

Words/ Documents	going	to	today	i	am	it	is	rain
Document 1	0	0.07	0.07	0	0	0.17	0.17	0.17
Document 2	0	0	0.07	0.07	0.07	0	0	0
Document 3	0	0.05	0	0.05	0.05	0	0	0

Remember, the final equation = TF-IDF = TF * IDF

Using this table, notice that words like ‘it’,’is’,’rain’ are important for document 1 but not for document 2 and document 3 which means Document 1 and 2&3 are different w.r.t talking about rain

https://medium.com/analytics-vidhya/tf-idf-term-frequency-technique-easiest-explanation-for-text-classification-in-nlp-with-code-8ca3912e58c3

NLP

- **Advancements in algorithms** that made training deep architectures more stable (e.g., better weight initialization techniques, normalization methods).
- **Increased computational power**, especially GPUs that can perform parallel computations efficiently. Things that used to take months, now can be done in days. Cloud Infrastructures.
- **Availability of large datasets which deep networks require to learn effectively**.

https://hackernoon.com/10-biggest-image-datasets-for-computer-vision

Src:Gareth James, D. W. (2023). An Introduction to Statistical Learning: with Applications in Python. Springer.
http://introtodeeplearning.com/

The NLP Process…

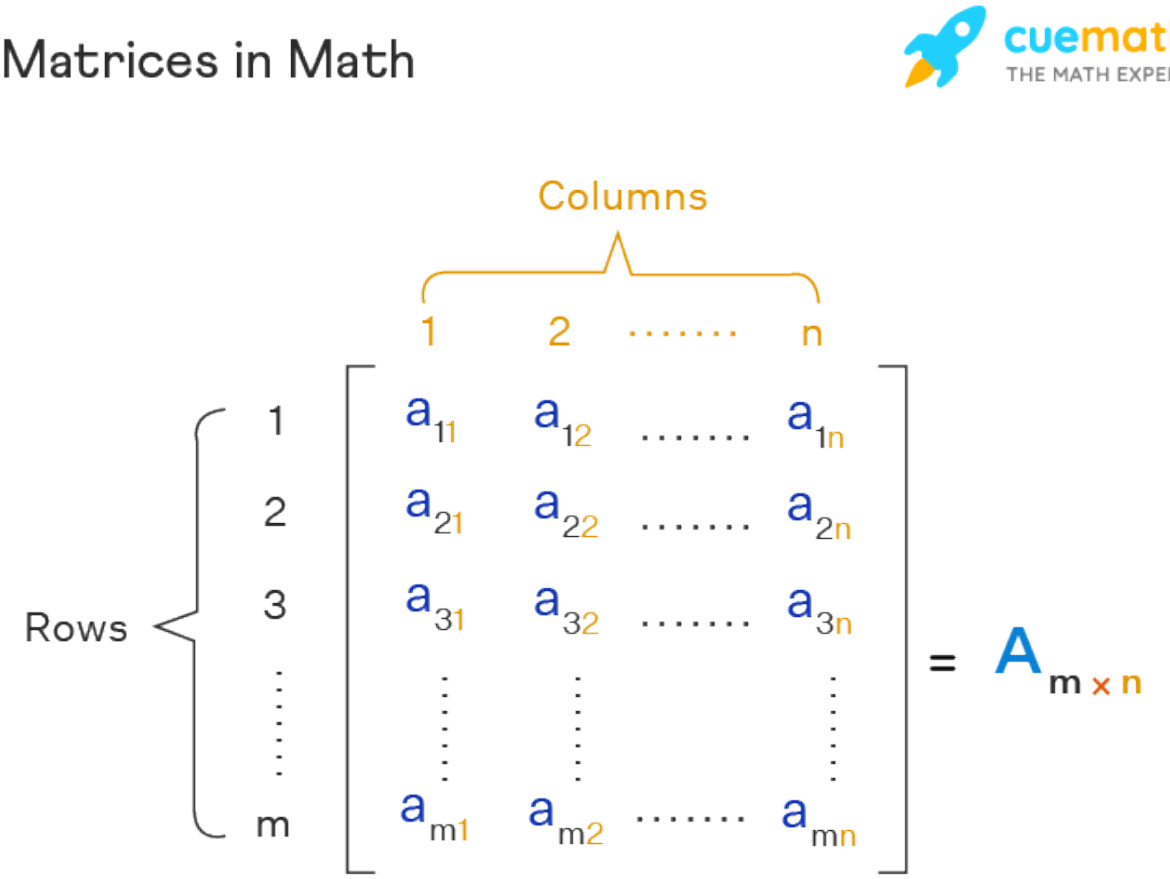
Words → Digitize → Feed to "Model" & Train → Test Model

Src:Gareth James, D. W. (2023). An Introduction to Statistical Learning: with Applications in Python. Springer.
https://en.wikipedia.org/wiki/Rule-based_machine_translation

At the core of all of this?

Hint…..

Matrix Operations



Src: https://www.cuemath.com/algebra/solve-matrices/

Tensors

(11)

SCALAR

5	3	7

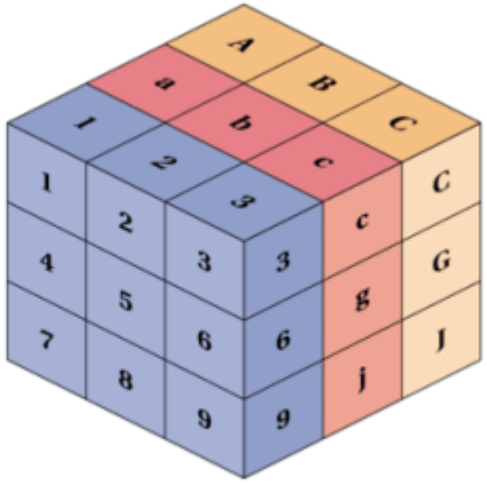
Row Vector
(shape 1x3)

5
1.5
2

Column Vector
(shape 3x1)

$$\begin{bmatrix} 4 & 19 & 8 \\ 16 & 3 & 5 \end{bmatrix}$$

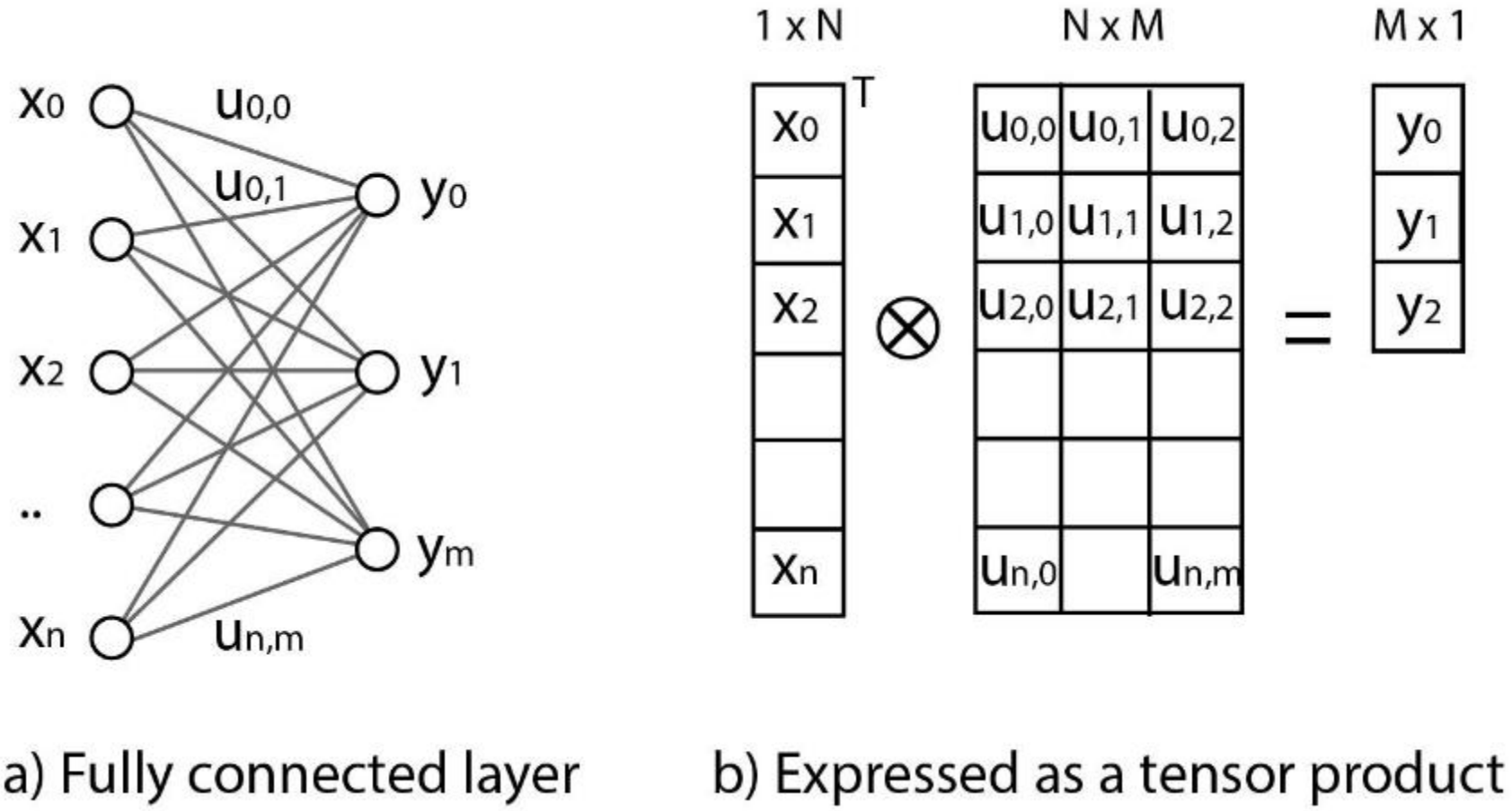
MATRIX



TENSOR

Src: https://www.cuemath.com/algebra/solve-matrices/

Tensors



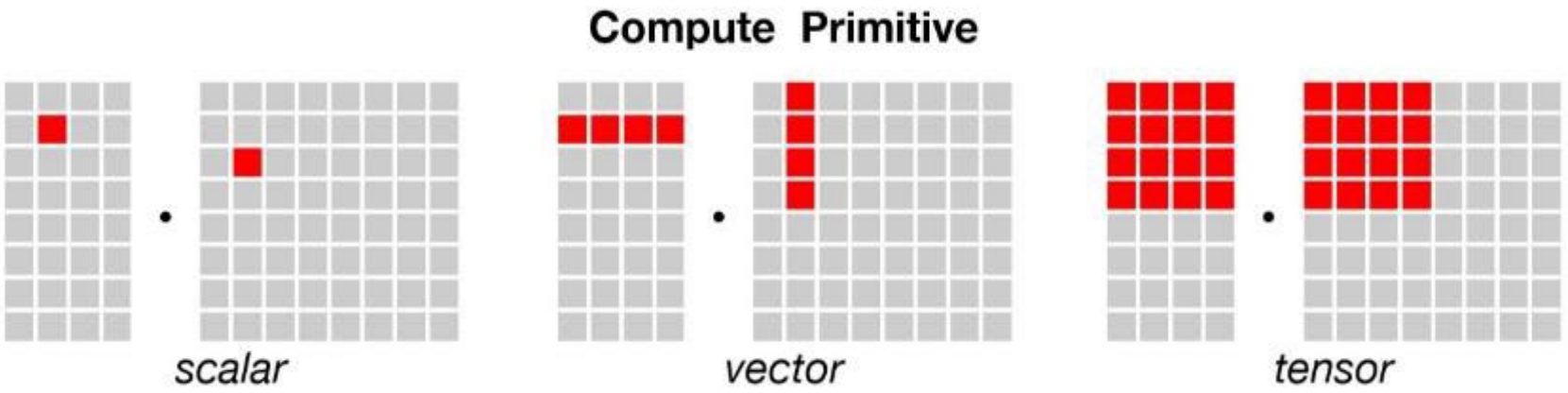
Src:https://commons.wikimedia.org/wiki/File:Fully_connected_neural_network_and_it%27s_expression_as_a_tensor_product.jpg

CPUs, GPUs, TPUs

Feature	CPU	GPU	TPU
Purpose	Multipurpose	Specialized for parallel computation	Special hardware for matrix processing
Latency for Arithmetic Operations	Very low	High compared to CPU	High
Throughput	Low - Not favorable for simultaneous arithmetic operations on many numbers	High - Can perform arithmetic operations simultaneously	Very High
Computation Method	Sequential - Performs computations one after the other	Parallel - Can compute in parallel	Extreme parallelism

Src:https://medium.com/dataseries/cpu-gpu-and-tpu-machine-learning-5033e25b8a0f

CPUs, GPUs, TPUs



The dimension of data are:

- CPU: 1 X 1 data unit
- GPU: 1 X N data unit
- TPU: N X N data unit

Src:https://iq.opengenus.org/cpu-vs-gpu-vs-tpu/

CPUs, GPUs, TPUs

Let's consider an example, multiplication of two matrices:

[1, 2, 3] * [4, 5, 6] = [4, 10, 18]

CPU does multiplication operation in just 2ns. where as GPU has 4ns for multiplication.

But CPU does this sequentially which takes

2 + 2 + 2 = 6ns

But GPU does this parallel processing that is multiplying each at same time. So it takes

4ns

Src:https://medium.com/dataseries/cpu-gpu-and-tpu-machine-learning-5033e25b8a0f

What is CUDA?

CUDA (Compute Unified Device Architecture) is a proprietary and closed-source parallel computing platform and application programming interface (API) that allows software to use certain types of graphics processing units (GPUs) for general-purpose processing, an approach called general-purpose computing on GPUs (GPGPU). CUDA is a software layer that gives direct access to the GPU's virtual instruction set and parallel computational elements for the execution of compute kernels

Src:https://en.wikipedia.org/wiki/CUDA

What is CUDA?

	Nvidia Geforce GTX 980 Ti	GTX Geforce GTX 1080
No. of Transistors	8,100,000,000	7,200,000,000
No. of CUDA cores	2816	2560
No. of Transistors/Core	2,876,420	2,812,500
Clock speed	1500 MHz	2000 MHz

Src:https://en.wikipedia.org/wiki/CUDA

more GPUs?

- GPU utilization does not scale as we add more of them. The more GPUs are in use, the more GPU-to-GPU communication and synchronization there is. As data is being transmitted across more GPUs, GPU utilization starts to drop due to idle time amid data transmission, making it more expensive to train the models

https://liu-gendary.medium.com/llm-explained-the-llm-training-landscape-82c803495caa

more GPUs?

- Taking GPT-3 as an example, the 175B version requires 3e23 FLOP (Floating Point Operation) of computation for training. If we are using the most available state-of-the-art GPU, Nvidia A100, which can conduct around 312 TeraFLOP per second (TFLOPs), a single GPU needs 30 years to train GPT-3 without factoring in actual utilization.

https://liu-gendary.medium.com/llm-explained-the-llm-training-landscape-82c803495caa

Hands On…..

Sample CNN - CPUs

19m

```
import tensorflow as tf
from tensorflow.keras.datasets import mnist
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Conv2D, Flatten, Dropout
from tensorflow.keras.utils import to_categorical

# Load and prepare the MNIST dataset
(x_train, y_train), (x_test, y_test) = mnist.load_data()
x_train, x_test = x_train / 255.0, x_test / 255.0  # Normalize to [0, 1]
x_train = x_train[..., tf.newaxis]
x_test = x_test[..., tf.newaxis]
y_train = to_categorical(y_train)
y_test = to_categorical(y_test)

# Define a simple CNN model
def create_model():
    model = Sequential([
        Conv2D(32, kernel_size=(3, 3), activation='relu', input_shape=(28, 28, 1)),
        Conv2D(64, (3, 3), activation='relu'),
        Flatten(),
```

✓ 19m 43s completed at 2:59 AM

Src:https://medium.com/dataseries/cpu-gpu-and-tpu-machine-learning-5033e25b8a0f

Sample CNN - GPUs

```python
import tensorflow as tf
from tensorflow.keras.datasets import mnist
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Conv2D, Flatten, Dropout
from tensorflow.keras.utils import to_categorical

# Load and prepare the MNIST dataset
(x_train, y_train), (x_test, y_test) = mnist.load_data()
x_train, x_test = x_train / 255.0, x_test / 255.0  # Normalize to [0, 1]
x_train = x_train[..., tf.newaxis]
x_test = x_test[..., tf.newaxis]
y_train = to_categorical(y_train)
y_test = to_categorical(y_test)

# Define a simple CNN model
def create_model():
    model = Sequential([
        Conv2D(32, kernel_size=(3, 3), activation='relu', input_shape=(28, 28, 1)),
        Conv2D(64, (3, 3), activation='relu'),
        Flatten(),
        Dense(128, activation='relu'),
```

✓ 40s completed at 3:07 AM

Sample CNN - TPUs

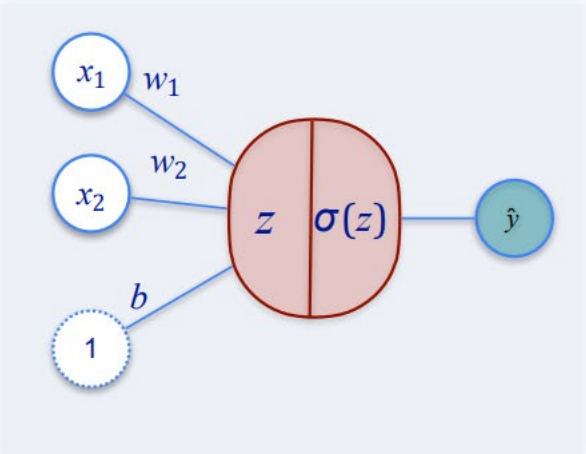
3m

```
import tensorflow as tf
from tensorflow.keras.datasets import mnist
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Conv2D, Flatten, Dropout
from tensorflow.keras.utils import to_categorical

# Load and prepare the MNIST dataset
(x_train, y_train), (x_test, y_test) = mnist.load_data()
x_train, x_test = x_train / 255.0, x_test / 255.0  # Normalize to [0, 1]
x_train = x_train[..., tf.newaxis]
x_test = x_test[..., tf.newaxis]
y_train = to_categorical(y_train)
y_test = to_categorical(y_test)

# Define a simple CNN model
def create_model():
    model = Sequential([
        Conv2D(32, kernel_size=(3, 3), activation='relu', input_shape=(28, 28, 1)),
        Conv2D(64, (3, 3), activation='relu'),
        Flatten(),
```

✓ 3m 7s completed at 3:12 AM

Sizing the Infrastructure - FLOPs

How many FLOPs do we need? Let us examine the calculations involved for 1 weight between two nodes in the ANN.



1

$L(y, \hat{y})$

0

To find optimal values for:
w_1, w_2, b

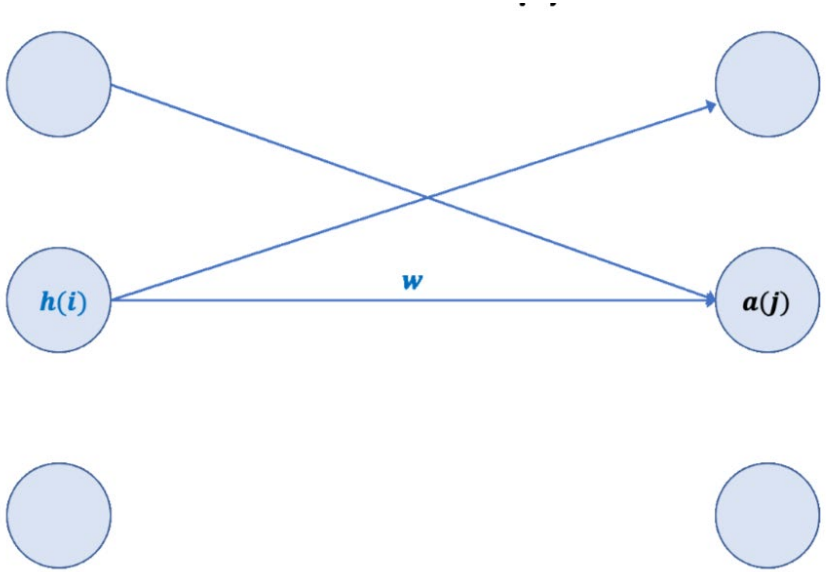
You need gradient descent

$$w_1 \rightarrow w_1 - \alpha(-x_1(y - \hat{y}))$$

$$w_2 \rightarrow w_2 - \alpha \frac{\partial L}{\partial w_2}$$

Sizing the Infrastructure - FLOPs

How many FLOPs do we need? Let us examine the calculations involved for 1 weight between two nodes in the ANN.



https://medium.com/@dzmitrybahdanau/the-flops-calculus-of-language-model-training-3b19c1f025e4

Sizing the Infrastructure - FLOPs

1. The unit i multiplies its output h(i) by w to send it to the unit j.
2. The unit j adds the unit i's contribution to its total input a(j).
3. **The unit j multiplies the incoming loss gradient dL/da(j) by w to send it back to the unit i.**
4. **The unit i adds the unit j's contribution to its total loss gradient dL/dh(i).**
5. The unit j multiplies its loss gradient dL/da(j) by the unit i's output h(i) to compute the loss gradient dL/dw for the given example.
6. The weight w adds the contribution from step 5 to its loss gradient accumulator dL/dw that aggregates gradients for all examples.

https://medium.com/@dzmitrybahdanau/the-flops-calculus-of-language-model-training-3b19c1f025e4

Sizing the Infrastructure - FLOPs

Transformer FLOPs Equation:

- the compute required to train a Transformer model (C)
- its number of parameters, or model size (N)
- the number of tokens that the model is trained on (D)

$$C \approx 6ND.$$

https://medium.com/@dzmitrybahdanau/the-flops-calculus-of-language-model-training-3b19c1f025e4

Sizing the Infrastructure - FLOPs

Another variant of the equation expresses the compute C as the product of cluster's throughput τ and training time T:

$$\tau T = C = 6ND.$$

Example: An 82B parameter Korean variant of GPT-3 called HyperCLOVA was trained on 150B tokens using a cluster of 1024 Nvidia A100 GPUs. How long could that take?

Solution: The peak float16 FLOPs throughput of A100 is τ = 312 teraFLOPs = 3.12e14 FLOPs. The total compute is C = 6 · 8.2e10 · 1.5e11 = 7.38e22. The training must have taken at least T = C / 1024τ / 86400 = 2.67 days.

Please note that this is a directional estimate of the capacity needs

https://medium.com/@dzmitrybahdanau/the-flops-calculus-of-language-model-training-3b19c1f025e4

What about Memory?

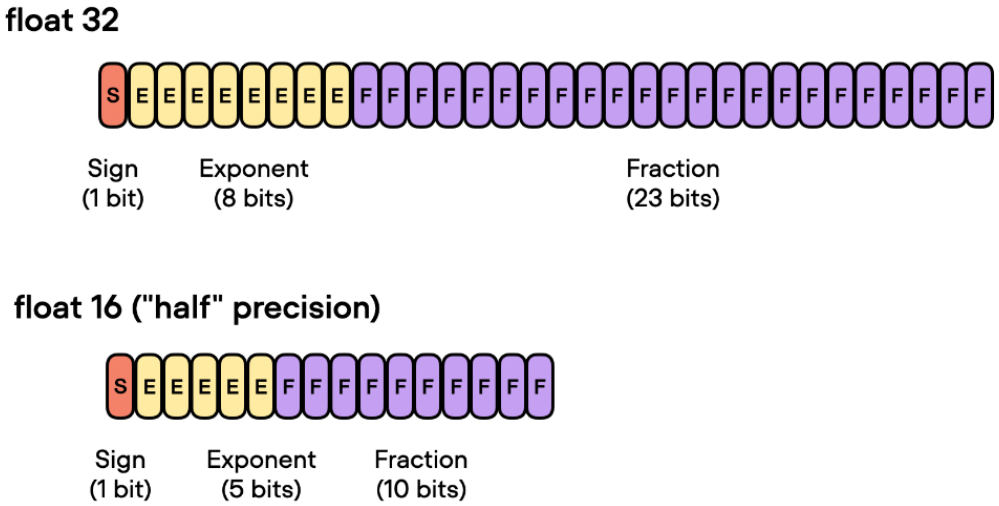
$$\text{Estimated GPU parameter Memory M} = \frac{(\text{\# of Parameters} * 4\text{B})}{32/\text{Q}} * 1.2$$

Q = The amount of bits that should be used for loading the model

Example:
Mistral 7B model
Memory needed for parameters = 7 * 4 * 1.2 = 33.2GB of memory.
In reality, to load the full model, you will need around 40-45GB.

Reducing “Resolution”

What if we do not have as much memory? → Reduce your “resolution”.



With FP16 precision, significantly reduces the model size and potentially improving inference speed on supported hardware, with a trade-off in precision that might be acceptable depending on the application.

Quantization - Introduction

- Quantization maps a large set of input values to a smaller set.
- It reduces memory usage and computational requirements.
- Commonly used in neural networks for deploying models on limited hardware.
- Allows larger models to run on devices with less memory.
- Introduces some loss in precision as a trade-off.

Quantization - Process

- First, identify the minimum and maximum values of the dataset.
- For 4-bit quantization, there are 16 levels ranging from -8 to 7.
- Calculate the scale factor to map the original range to the quantization range.
- Convert original values to the nearest quantized values.
- Convert quantized values back to floating-point for computation.
- Quantized values are integers, and clamping may occur to stay within range

Quantization - Explanation

- Quantization is used to store data efficiently with fewer bits.
- During computation, higher precision is needed for accurate results.
- Dequantization converts low-precision values back to higher precision.
- Example: 4-bit values dequantized to 16-bit floating-point for calculations.
- Allows the use of efficient storage and precise computations.
- Ensures balance between memory savings and computational accuracy.

Quantization - Example

- Given weights: [0.02345, -0.56789, 1.23456, -0.34567].
- Min and max values are -0.56789 and 1.23456, respectively.
- Calculate the range: 1.23456 - (-0.56789) = 1.80245.
- Quantization levels for 4-bit signed integers are from -8 to 7.
- Compute the scale factor: 15 / 1.80245 ≈ 8.3205.
- Apply the scale factor to quantize the values.

Quantization - Example

- For 0.02345, multiply by 8.3205 to get 0.195; round to 0.
- For -0.56789, multiply by 8.3205 to get -4.723; round to -5.
- For 1.23456, multiply by 8.3205 to get 10.272; round to 10 (clamped to 7).
- For -0.34567, multiply by 8.3205 to get -2.876; round to -3.
- The quantized values are [0, -5, 7, -3].
- These values represent the nearest integers within the quantized range.

Example: De-Quantization

- Convert back to floating-point using the scale factor.
- For 0, divide by 8.3205 to get 0.0.
- For -5, divide by 8.3205 to get approximately -0.6006.
- For 7, divide by 8.3205 to get approximately 0.8411.
- For -3, divide by 8.3205 to get approximately -0.3603.
- Dequantized values approximate the original values.
- .

Quantization: Key Take-aways

- Quantization reduces model size, making it memory efficient.
- Dequantization approximates the original values for computation.
- Clamping ensures values stay within the representable range.
- The trade-off involves balancing memory savings with precision loss.
- Enables the deployment of large models on resource-constrained hardware.
- .

Key Libraries:

NLTK: CPU-based Standalone NLP Library: NLTK (Natural Language Toolkit) is indeed more of a CPU-based standalone library focused on NLP. It's one of the oldest and most comprehensive libraries for text processing and linguistic analysis, offering a wide range of functionalities for tokenization, parsing, tagging, and semantic reasoning, among others. NLTK is particularly well-suited for educational purposes and prototyping, where the primary goal is to learn NLP concepts or build simple NLP applications without the need for the computational power provided by GPUs or TPUs.

Key Libraries:

NLTK: https://www.nltk.org/api/nltk.html

Book: https://www.nltk.org/book/

There is also spacy: https://spacy.io/models

And many more am sure….

Key Libraries:

PyTorch and TensorFlow

Integration with NLP Libraries: PyTorch and TensorFlow are two of the most popular deep learning frameworks, and they serve as the backbone for many NLP libraries. Both PyTorch and TensorFlow can be used independently for a wide range of deep learning tasks, including NLP.

https://pytorch.org/join

https://www.tensorflow.org/

Recurrent NNs:

Many data sources are sequential in nature, and call for special treatment when building predictive models. Examples include:

- Documents such as book and movie reviews, newspaper articles, and tweets. The sequence and relative positions of words in a document capture the narrative, theme and tone, and can be exploited in tasks such as topic classification, sentiment analysis, and language translation.
- •Time series of temperature, rainfall, wind speed, air quality, and so on. We may want to forecast the weather several days ahead, or climate several decades ahead.

Src:Gareth James, D. W. (2023). An Introduction to Statistical Learning: with Applications in Python. Springer.
http://introtodeeplearning.com/

Recurrent NNs:

- Financial time series, where we track market indices, trading volumes,stock and bond prices, and exchange rates. Here prediction is often difficult, but as we will see, certain indices can be predicted with reasonable accuracy.
- Recorded speech, musical recordings, and other sound recordings. We may want to give a text transcription of a speech, or perhaps a language translation. We may want to assess the quality of a piece of music, or assign certain attributes.
- Handwriting, such as doctor's notes, and handwritten digits such as zip codes. Here we want to turn the handwriting into digital text, or read the digits (optical character recognition).

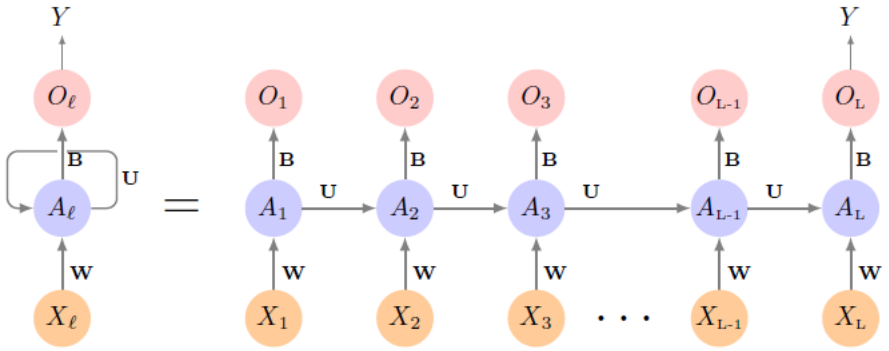
Src:Gareth James, D. W. (2023). An Introduction to Statistical Learning: with Applications in Python. Springer.
http://introtodeeplearning.com/

Recurrent NNs:

- recurrent neural network (RNN), the input object X is a sequence. Consider a corpus of documents, such as the collection of IMDb movie reviews.
- Each document can be represented as a sequence of L words, so X = {X1,X2, . . . ,XL}, where each X$ represents a word. The order of the words, and closeness of certain words in a sentence, convey semantic meaning. RNNs are designed to accommodate and take advantage of the sequential nature of such input objects, much like convolutional neural networks accommodate the spatial structure of image inputs. The output Y can also be a sequence (such as in language translation), but often is a scalar, like the binary sentiment label of a movie review document.

Src:Gareth James, D. W. (2023). An Introduction to Statistical Learning: with Applications in Python. Springer.
http://introtodeeplearning.com/

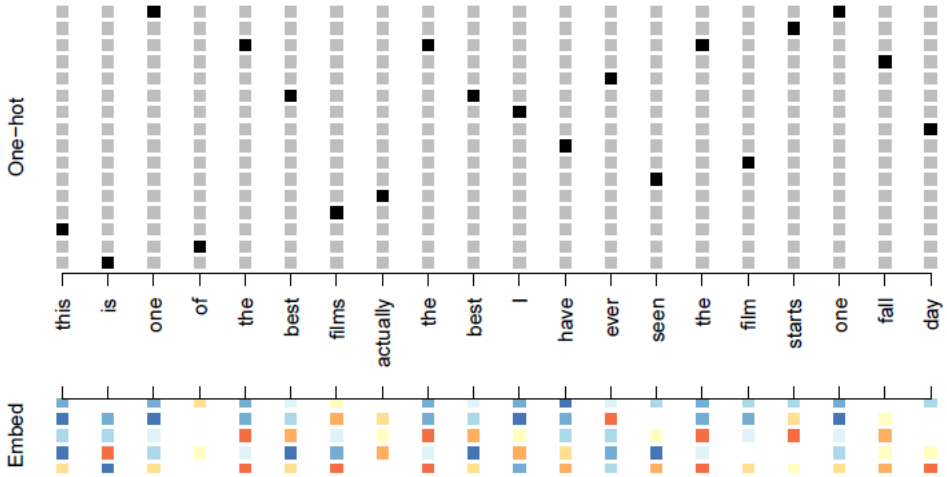
Recurrent NNs:



Schematic of a simple recurrent neural network. The input is a sequence of vectors {X"}L1 , and here the target is a single response. The network processes the input sequence X sequentially; each X" feeds into the hidden layer, which also has as input the activation vector A"−1 from the previous element in the sequence, and produces the current activation vector A". The same collections of weights W, U and B are used as each element of the sequence is processed. The output layer produces a sequence of predictions O" from the current activation A", but typically only the last of these, OL, is of relevance. To the left of the equal sign is a concise representation of the network, which is unrolled into a more explicit version on the right.

Src:Gareth James, D. W. (2023). An Introduction to Statistical Learning: with Applications in Python. Springer.
http://introtodeeplearning.com/

Recurrent NNs & Embeddings:



- Depiction of a sequence of 20 words representing a single document: one-hot encoded using a dictionary of 16 words and embedded in an m-dimensional space with m = 5 (bottom panel).
- Where does E (Embeddings) come from? If we have a large corpus of labeled documents, we can have the neural network learn E as part of the optimization. In this case E is referred to as an embedding layer.
- Two pretrained embeddings, word2vec and GloVe, are widely used

Src:Gareth James, D. W. (2023). An Introduction to Statistical Learning: with Applications in Python. Springer.
http://introtodeeplearning.com/

Recurrent NNs Variant - LSTM:

- The model was trained with dropout regularization on the 25,000 reviews in the designated training set, and achieved a 76% accuracy
- Long term and short term memory (LSTM). Two tracks of hidden-layer activations are maintained, so that when the activation A$ is computed, it gets input from hidden units both further back in time, and closer in time — a so-called LSTM RNN.
- With long sequences, this overcomes the problem of early signals being washed out by the time they get propagated through the chain to the final activation vector AL
- When we refit our model using the LSTM architecture for the hidden layer, the performance improved to 87%.

Src:Gareth James, D. W. (2023). An Introduction to Statistical Learning: with Applications in Python. Springer.
http://introtodeeplearning.com/

Recurrent NNs & Document Sentiment:

- IMDB Reviews
- Each word in our document is represented by a one-hot-encoded vector (dummy variable) with 10,000 elements (one per word in the dictionary)! An approach that has become popular is to represent each word in a much lower-dimensional embedding space. This means that rather than representing each word by a binary embedding vector with 9,999 zeros and a single one in some position, we will represent it instead by a set of m real numbers, none of which are typically zero.
- Each document is now represented as a sequence of mvectors that represents the sequence of words. The next step is to limit each document to the last L words. Documents that are shorter than L get padded with zeros upfront. So now each document is represented by a series consisting of L vectors X = {X1,X2, . . . ,XL}, and each X_L in the sequence has m components.
- We now use the RNN structure in Figure 10.12. The training corpus consists of n separate series (documents) of length L, each of which gets processed sequentially from left to right. In the process$_L$, a parallel series of hidden activation vectors A$, % = 1, . . . ,L is created for each document. A feeds into the output layer to produce the evolving prediction O_L. We use the final value OL to predict the response: the sentiment of the review.

Src:Gareth James, D. W. (2023). An Introduction to Statistical Learning: with Applications in Python. Springer.
http://introtodeeplearning.com/

RNNs & Time Series Forecasting

- Sequential Data Processing: RNNs are inherently designed for sequential data. A time series is a sequence where a metric is recorded at regular time intervals. RNNs can process this data in its natural form, without the need for complex transformations.
- Memory of Past Inputs: The fundamental feature of RNNs is their internal memory. This memory allows RNNs to remember previous inputs in the sequence and use this memory to influence the current output. For time series forecasting, this is crucial since past data points often influence future ones.
- Vanishing and Exploding Gradient Problem: Traditional RNNs suffer from the vanishing and exploding gradient problems, which can make them challenging to train on long sequences. This issue means that they might not always be the best choice for time series with long dependencies → **LSTM.**

Src:Gareth James, D. W. (2023). An Introduction to Statistical Learning: with Applications in Python. Springer.
http://introtodeeplearning.com/

RNNs in NLP

They were taken over by Transformers

Src:Gareth James, D. W. (2023). An Introduction to Statistical Learning: with Applications in Python. Springer.
http://introtodeeplearning.com/

Midterm SEAS8520

- Please check that the Honorlock system is working for you.
- Please read the questions carefully and keep the focus to the question boundaries (25-30 questions).
- If a question has more than one correct answer, pick the answer that combines these two.
- Please spend the time going thru the questions. You do not need the 3 hours, but the test needs more than 30 minutes.
- You will get the score when you submit. Details after due date. Course scores will be curved at the end by the department.
- Good Luck in the Midterm

BACK-UP

Hands On…..

References

In addition to the references in each slide the below are leveraged throughout this course.

Gareth James, D. W. (2023). An Introduction to Statistical Learning: with Applications in Python. Springer.

VanderPlas, J. (2017). Python Data Science Handbook. O'Reilly.
Wolff, S. G. (2018). Less is more: optimizing classification performance through feature selection in a very-high-resolution remote sensing object-based urban application. GIScience & Remote Sensing. doi:10.1080/15481603.2017.1408892

Prince, S. J. (January 28, 2024). Understanding Deep Learning. MIT Press.
Chollet, F. (2021). Deep Learning with Python, Second Edition (2nd ed.). Manning