NYU FRE 7773 - Week 7

Machine Learning in Financial Engineering

Ethan Rosenthal

Intro to Deep Learning

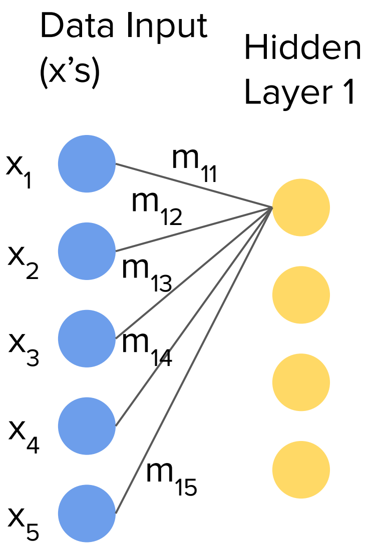
Machine Learning in Financial Engineering

Ethan Rosenthal

Artificial Neural Networks

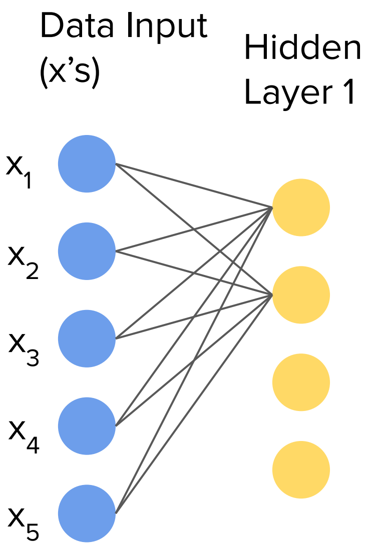
Feed Forward Network / Multilayer Perceptron

Feed Forward Network / Multilayer Perceptron



$$h_1 = f(b^{(1)}_1 + m^{(1)}_{11} * x_1 + m^{(1)}_{12} * x_2 + \ldots)$$

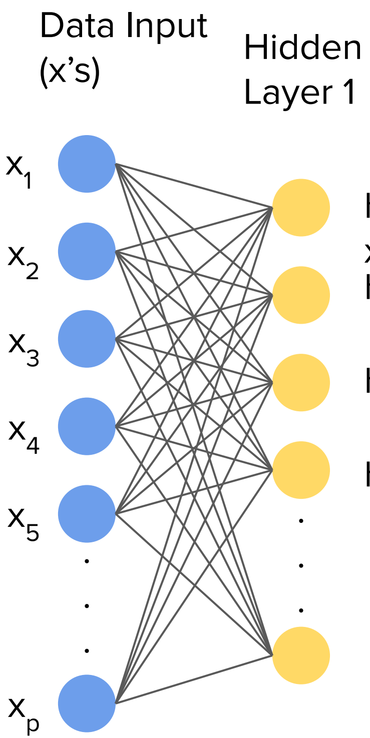
Feed Forward Network / Multilayer Perceptron



$$h_1 = f(b^{(1)}_1 + m^{(1)}_{11} * x_1 + m^{(1)}_{12} * x_2 + \ldots)$$

$$h_2 = f(b^{(1)}_2 + m^{(1)}_{21} * x_1 + m^{(1)}_{22} * x_2 + \ldots)$$

Feed Forward Network / Multilayer Perceptron



$$h^{(1)}_1 = f(b^{(1)}_1 + m^{(1)}_{11}{}^{*}x_1 + m^{(1)}_{12}{}^{*}\, x_2 +\ldots\, m^{(1)}_{1p}{}^{*}\, x_p)$$

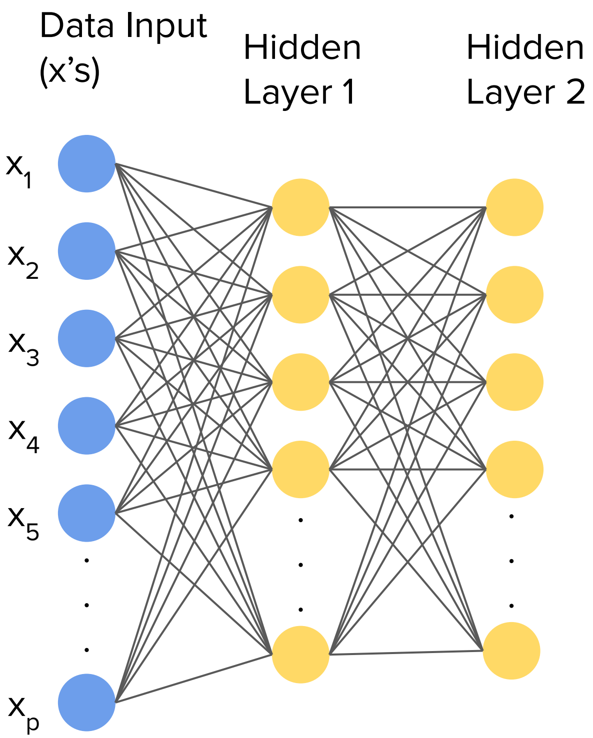
$$h^{(1)}_2 = f(b^{(1)}_2 + m^{(1)}_{21}{}^{*}x_1 + m^{(1)}_{22}{}^{*}\, x_2 +\ldots\, m^{(1)}_{2p}{}^{*}\, x_p)$$

$$h^{(1)}_3 = f(b^{(1)}_3 + m^{(1)}_{31}{}^{*}x_1 + m^{(1)}_{32}{}^{*}\, x_2 +\ldots\, m^{(1)}_{3p}{}^{*}\, x_p)$$

$$h^{(1)}_4 = f(b^{(1)}_4 + m^{(1)}_{41}{}^{*}x_1 + m^{(1)}_{42}{}^{*}\, x_2 +\ldots\, m^{(1)}_{4p}{}^{*}\, x_p)$$

$$h^{(1)}_{L(1)} = f(b^{(1)}_{L(1)} + m^{(1)}_{L(1)1}{}^{*}x_1 + m^{(1)}_{L(1)2}{}^{*}\, x_2 +\ldots\, m^{(1)}_{L(1)p}{}^{*}\, x_p)$$

Feed Forward Network / Multilayer Perceptron



$$h^{(2)}_1 = f(b^{(2)}_1 + m^{(2)}_{11} * h^{(1)}_1 + m^{(2)}_{12} * h^{(1)}_2 + \dots m^{(2)}_{1L(1)} * h^{(1)}_{L(1)})$$

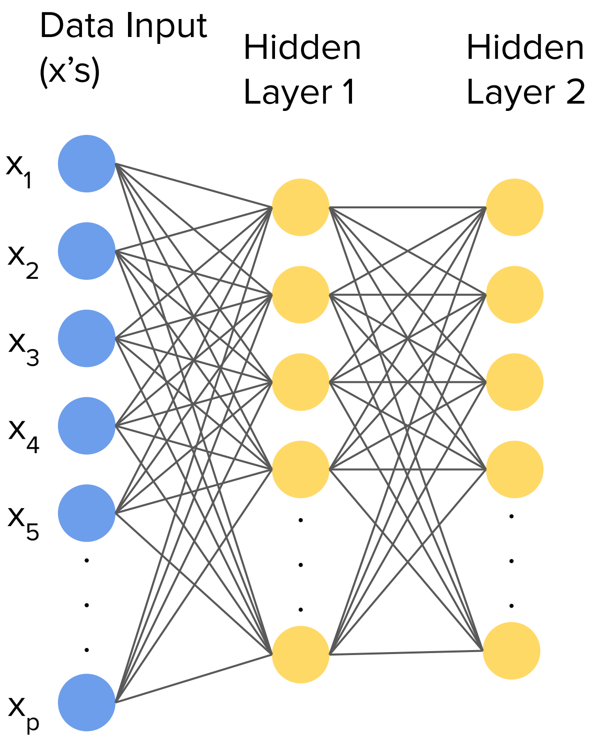
$$h^{(2)}_2 = f(b^{(2)}_2 + m^{(2)}_{21} * h^{(1)}_1 + m^{(2)}_{22} * h^{(1)}_2 + \dots m^{(2)}_{2L(1)} * h^{(1)}_{L(1)})$$

$$h^{(2)}_3 = f(b^{(2)}_3 + m^{(2)}_{31} * h^{(1)}_1 + m^{(2)}_{32} * h^{(1)}_2 + \dots m^{(2)}_{3L(1)} * h^{(1)}_{L(1)})$$

$$h^{(2)}_4 = f(b^{(2)}_4 + m^{(2)}_{41} * h^{(1)}_1 + m^{(2)}_{42} * h^{(1)}_2 + \dots m^{(2)}_{4L(1)} * h^{(1)}_{L(1)})$$

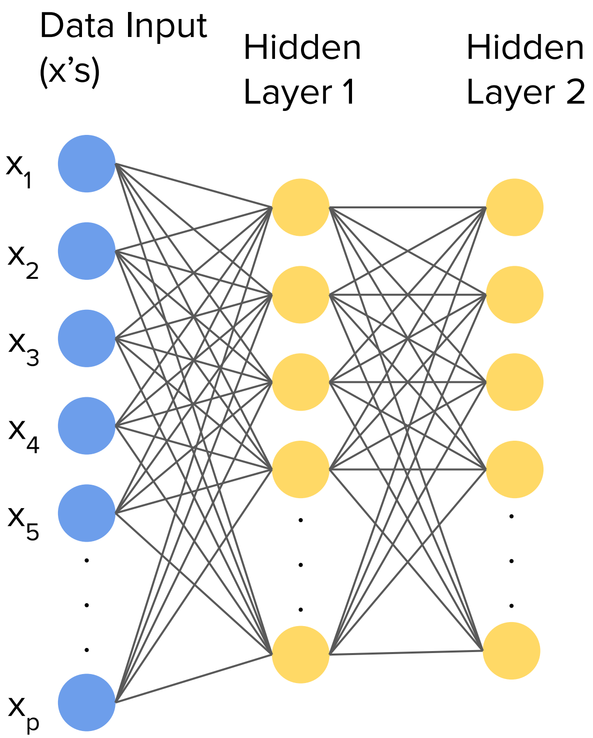
$$h^{(2)}_{L(2)} = f(b^{(2)}_{L(2)} + m^{(2)}_{L(2)1} * h^{(1)}_1 + m^{(2)}_{L(2)2} * h^{(1)}_2 + \dots m^{(2)}_{L(2)L(1)} * h^{(1)}_{L(1)})$$

Feed Forward Network / Multilayer Perceptron



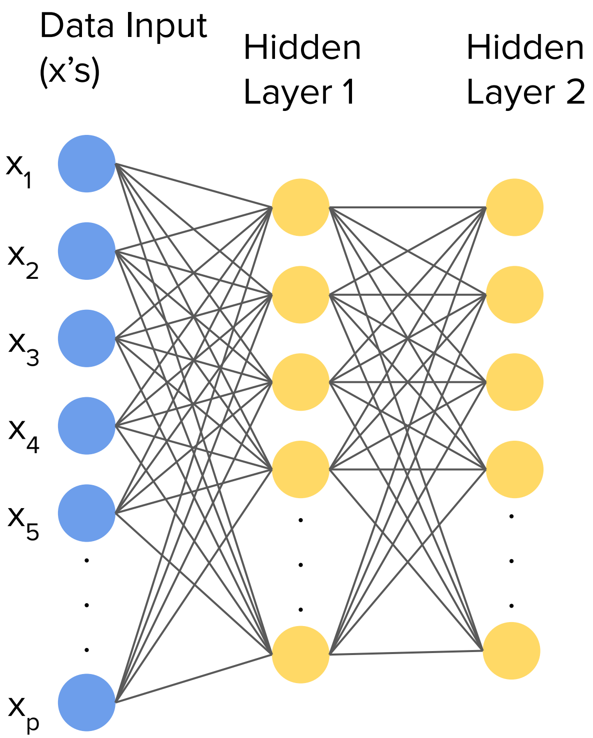
$h^{(2)}_1 = f(b^{(2)}_1 + m^{(2)}_{11} * \mathbf{h^{(1)}_1} + m^{(2)}_{12} * \mathbf{h^{(1)}_2} + \ldots\ m^{(2)}_{1L(1)} * \mathbf{h^{(1)}}_{L(1)})$

Feed Forward Network / Multilayer Perceptron



$$
\begin{aligned}
h^{(2)}_1 &= f(b^{(2)}_1 + m^{(2)}_{11} * \mathbf{h}^{(1)}_{1} + m^{(2)}_{12} * \mathbf{h}^{(1)}_{2} + \dots\ m^{(2)}_{1L(1)} * \mathbf{h}^{(1)}_{L(1)}\ ) \\
&= f(b^{(2)}_1 + \\
&\quad m^{(2)}_{11} * f(b^{(1)}_1 + m^{(1)}_{11} * x_1 + m^{(1)}_{12} * x_2 + \dots\ m^{(1)}_{1p} * x_p) \\
&\quad + m^{(2)}_{12} * f(b^{(1)}_2 + m^{(1)}_{21} * x_1 + m^{(1)}_{22} * x_2 + \dots\ m^{(1)}_{2p} * x_p) \\
&\quad + \dots \\
&\quad m^{(2)}_{1L(1)} * f(b^{(1)}_{L(1)} + m^{(1)}_{L(1)1} * x_1 + m^{(1)}_{L(1)2} * x_2 + \dots\ m^{(1)}_{L(1)p} * x_p) \\
&\quad)
\end{aligned}
$$

Feed Forward Network / Multilayer Perceptron

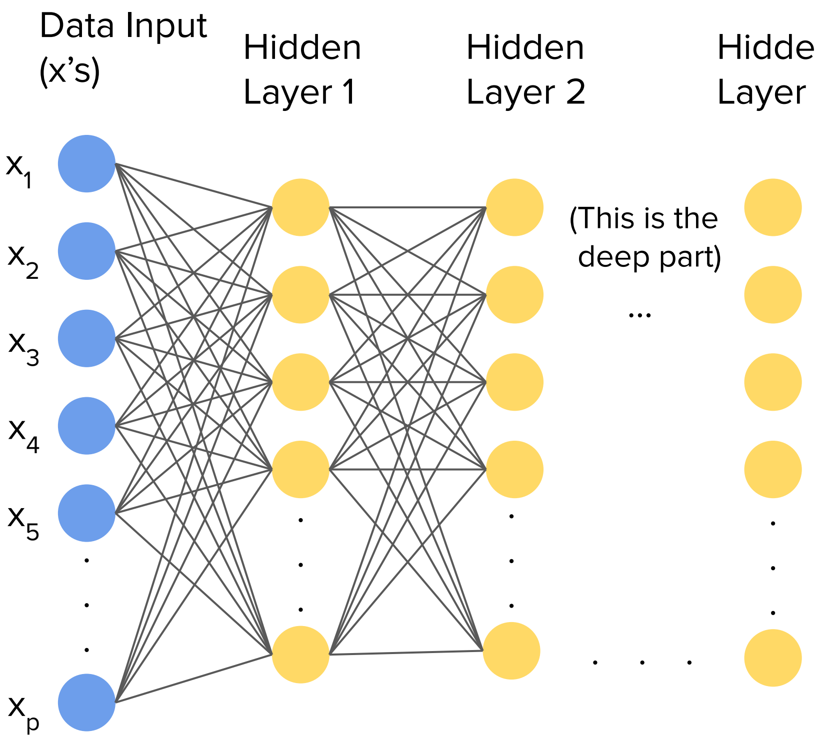


$$
\begin{aligned}
h^{(2)}_1 &= f(b^{(2)}_1 + m^{(2)}_{11} * \mathbf{h^{(1)}_1} + m^{(2)}_{12} * \mathbf{h^{(1)}_2} + \ldots\ m^{(2)}_{1L(1)} * \mathbf{h^{(1)}_{L(1)}}\ ) \\
&= f(b^{(2)}_1 + \\
&\quad m^{(2)}_{11} * f(b^{(1)}_1 + m^{(1)}_{11} * x_1 + m^{(1)}_{12} * x_2 + \ldots\ m^{(1)}_{1p} * x_p) \\
&\quad + m^{(2)}_{12} * f(b^{(1)}_2 + m^{(1)}_{21} * x_1 + m^{(1)}_{22} * x_2 + \ldots\ m^{(1)}_{2p} * x_p) \\
&\quad + \ldots \\
&\quad m^{(2)}_{1L(1)} * f(b^{(1)}_{L(1)} + m^{(1)}_{L(1)1} * x_1 + m^{(1)}_{L(1)2} * x_2 + \ldots\ m^{(1)}_{L(1)p} * x_p) \\
&\quad)
\end{aligned}
$$

$$h^{(2)}_1 = f(b^{(2)}_1; m_1^{(2)}; h^{(1)}) \quad , \quad h^{(1)}_1 = f(b^{(1)}_1; m_1^{(1)}; x)$$

$$h^{(2)}_1 = f(b^{(2)}_1; m_1^{(2)}; f(b^{(1)}_1; m_1^{(1)}; x))$$

Feed Forward Network / Multilayer Perceptron



$$h^{(K)}_1 = f(b^{(K)}_1 + m^{(K)}_{11} * h^{(K-1)}_1 + m^{(K)}_{12} * h^{(K-1)}_2 + \ldots m^{(K)}_{1L(K-1)} * h^{(K-1)}_{L(K-1)}$$

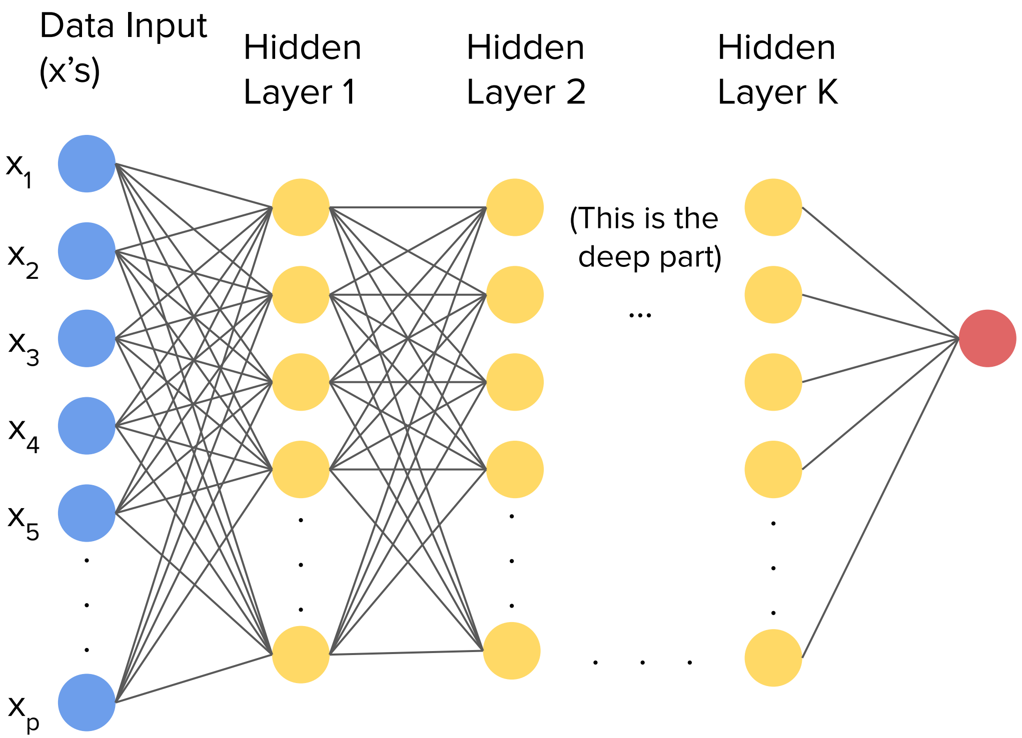
$$h^{(K)}_2 = f(b^{(K)}_2 + m^{(K)}_{21} * h^{(K-1)}_1 + m^{(K)}_{22} * h^{(K-1)}_2 + \ldots m^{(K)}_{2L(K-1)} * h^{(K-1)}_{L(K-1)}$$

$$h^{(K)}_3 = f(b^{(K)}_3 + m^{(K)}_{31} * h^{(K-1)}_1 + m^{(K)}_{32} * h^{(K-1)}_2 + \ldots m^{(K)}_{3L(K-1)} * h^{(K-1)}_{L(K-1)}$$

$$h^{(K)}_4 = f(b^{(K)}_4 + m^{(K)}_{41} * h^{(K-1)}_1 + m^{(K)}_{42} * h^{(K-1)}_2 + \ldots m^{(K)}_{4L(K-1)} * h^{(K-1)}_{L(K-1)}$$

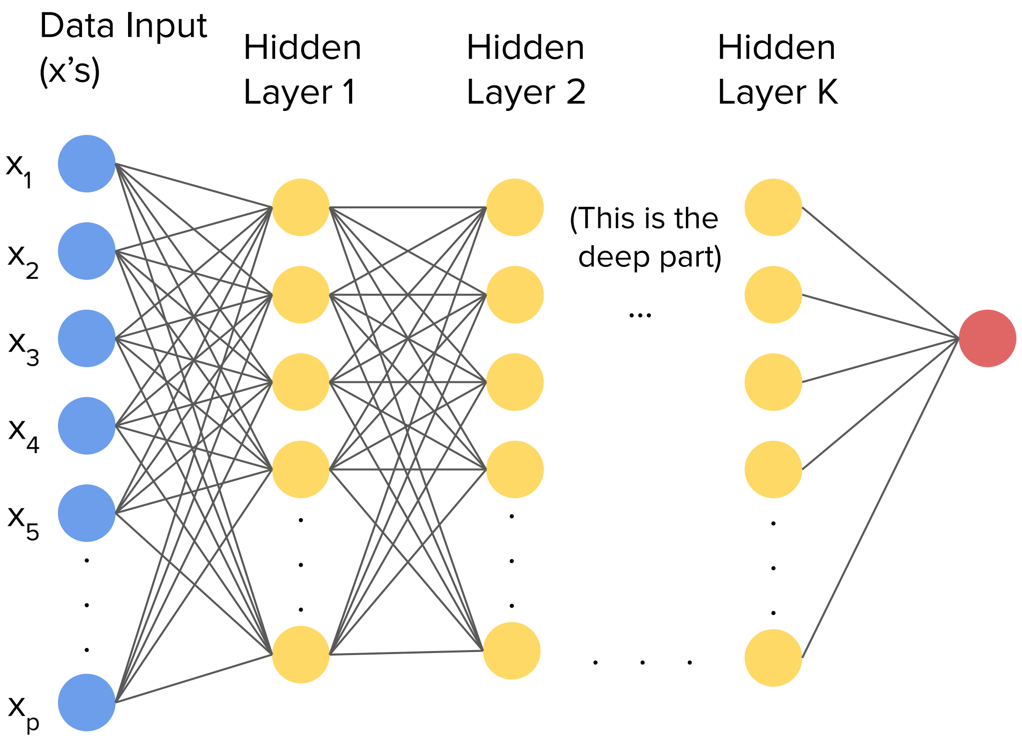
$$h^{(K)}_{L(K)} = f(b^{(K)}_{L(K)} + m^{(K)}_{L(K)1} * h^{(K-1)}_1 + m^{(K)}_{L(K)2} * h^{(K-1)}_2 + \ldots m^{(K)}_{L(K)L(K-1)} * h^{(K-1)}_{L(K-1)})$$

Feed Forward Network / Multilayer Perceptron



$$y = b^{(K)}_{1} + m^{(K+1)}_{11} {}^{*}h^{(K)}_{1} + m^{(K+1)}_{12} {}^{*}h^{(K)}_{2} + \ldots + m^{(K+1)}_{1L(K)} {}^{*}h^{(K)}_{L(K)}$$

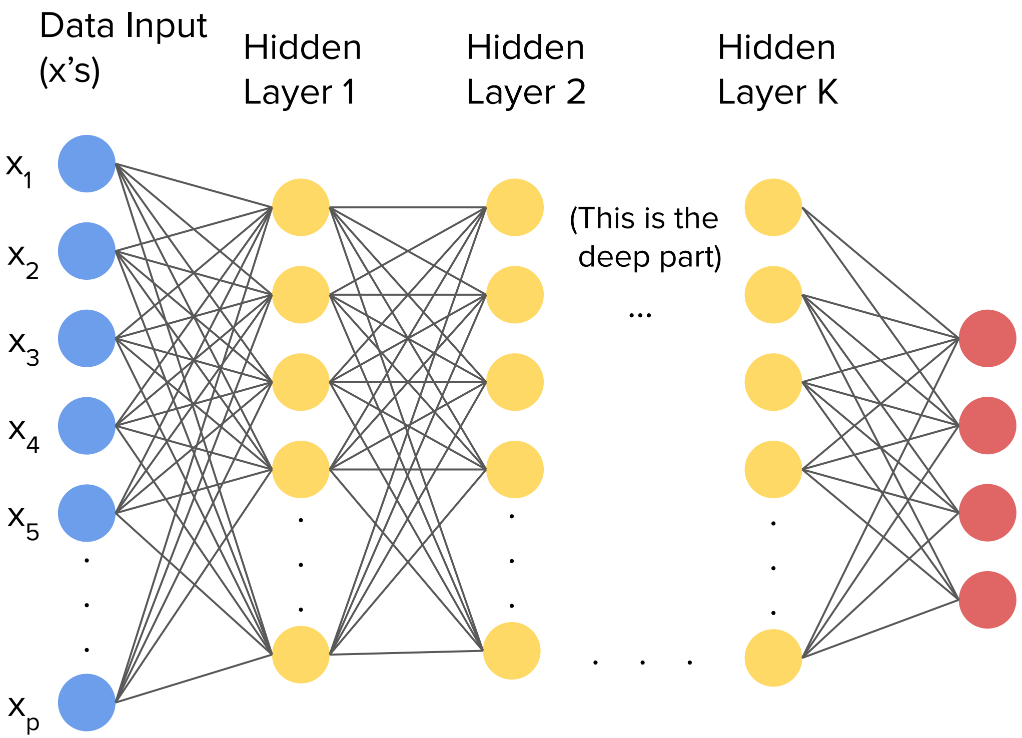
Feed Forward Network – Binary Classification



$$y = \sigma(b^{(K)}_1 + m^{(K+1)}_{11} * h^{(K)}_1 + m^{(K+1)}_{12} * h^{(K)}_2 + \ldots + m^{(K+1)}_{1L(K)} * h^{(K)}_{L(K)})$$

$$\sigma(x) = 1 / (1 + e^{-x})$$

Feed Forward Network – Multiclass Classification

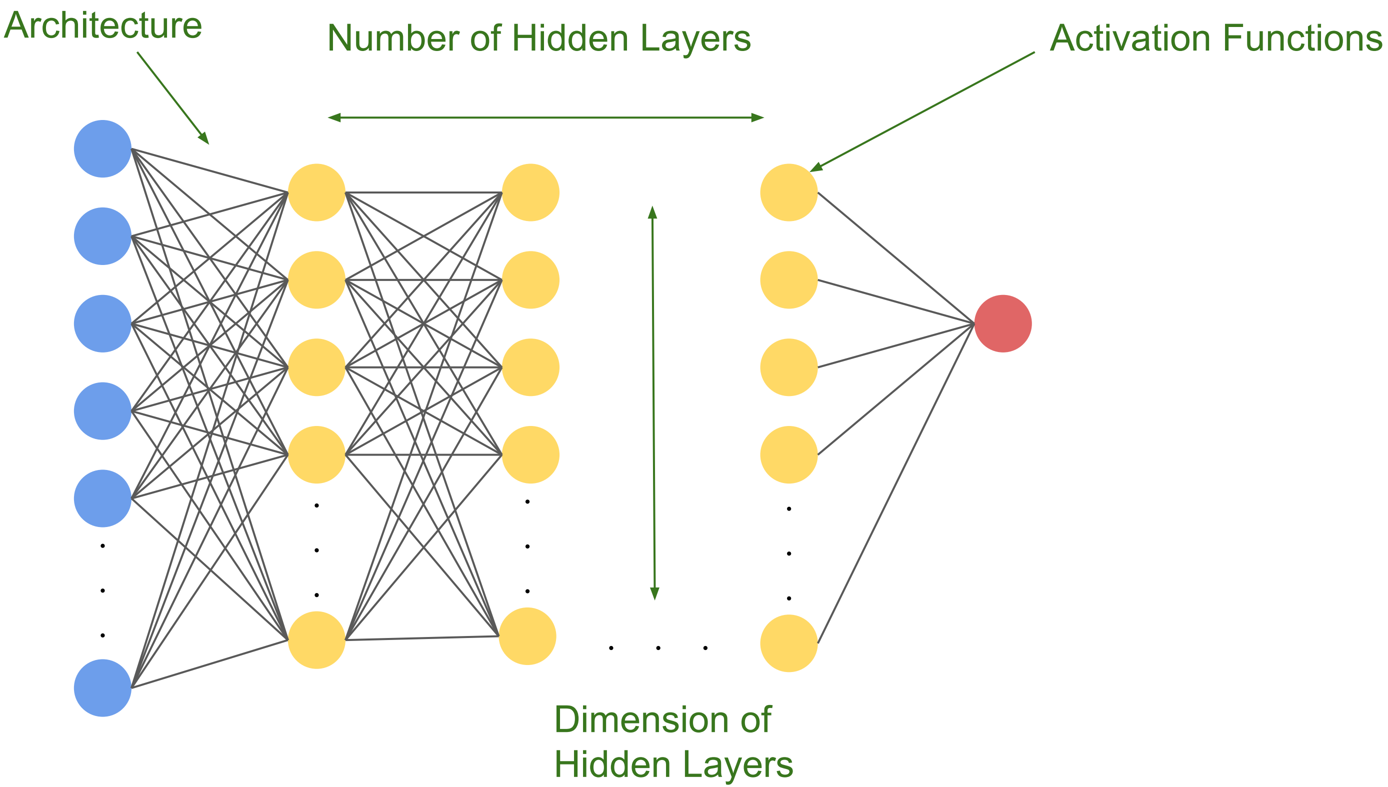


$$y_1 = \sigma(b^{(K)}_1 + m^{(K+1)}_{11} * h^{(K)}_1 + m^{(K+1)}_{12} * h^{(K)}_2 +$$

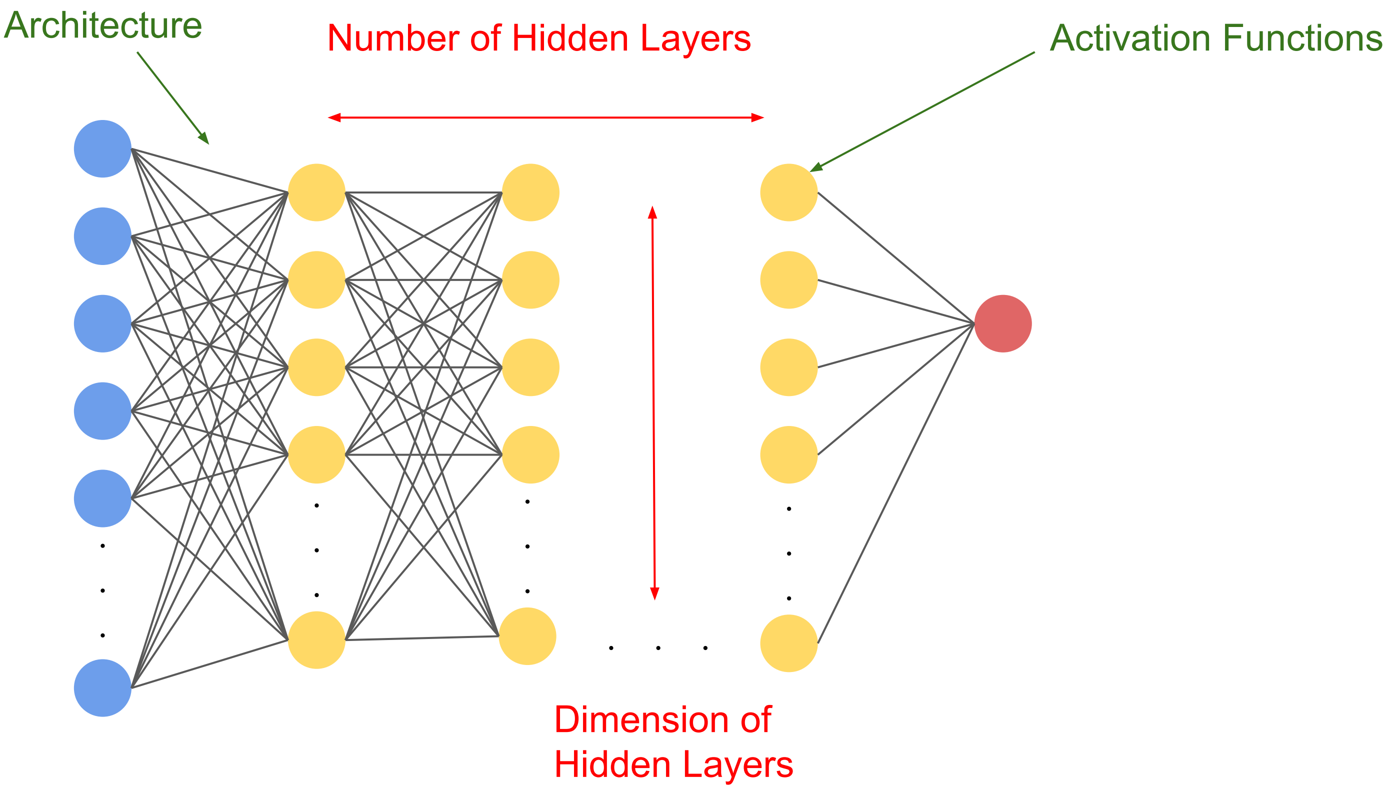
$$\ldots + m^{(K+1)}_{1L(K)} * h^{(K)}_{L(K)})$$

$$s(x_1) = e^{-x1} / \sum_i e^{-xi}$$

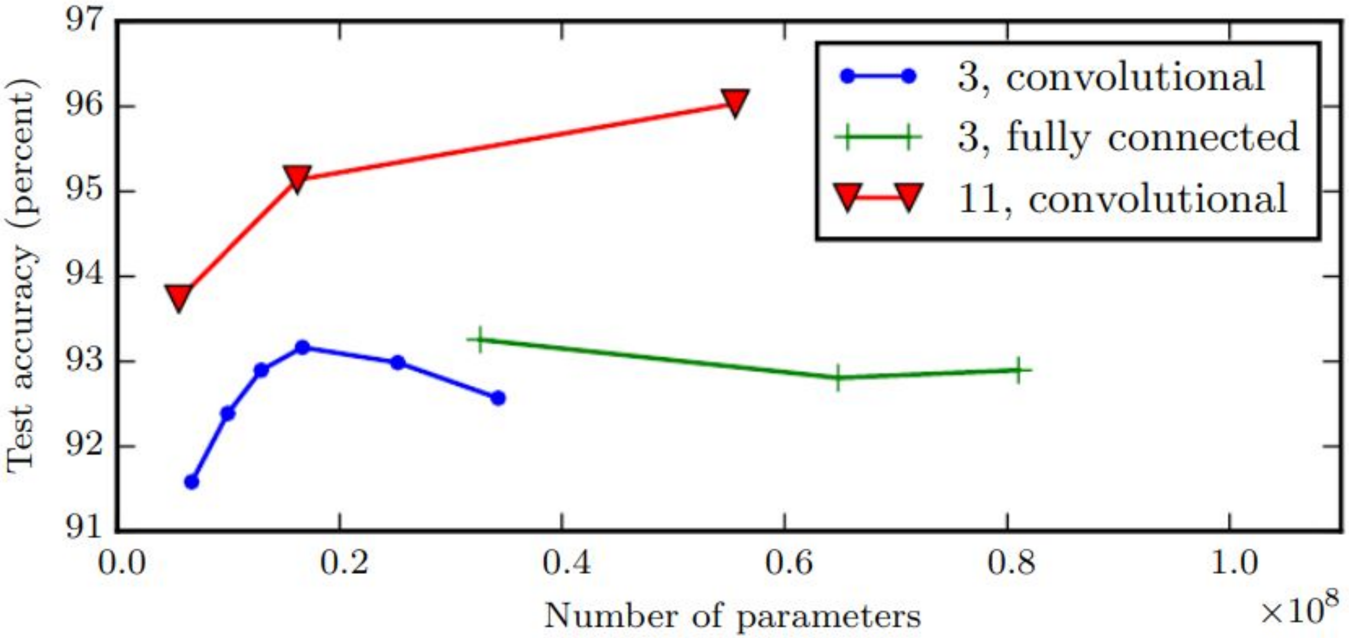
Knobs at our disposal



Knobs at our disposal

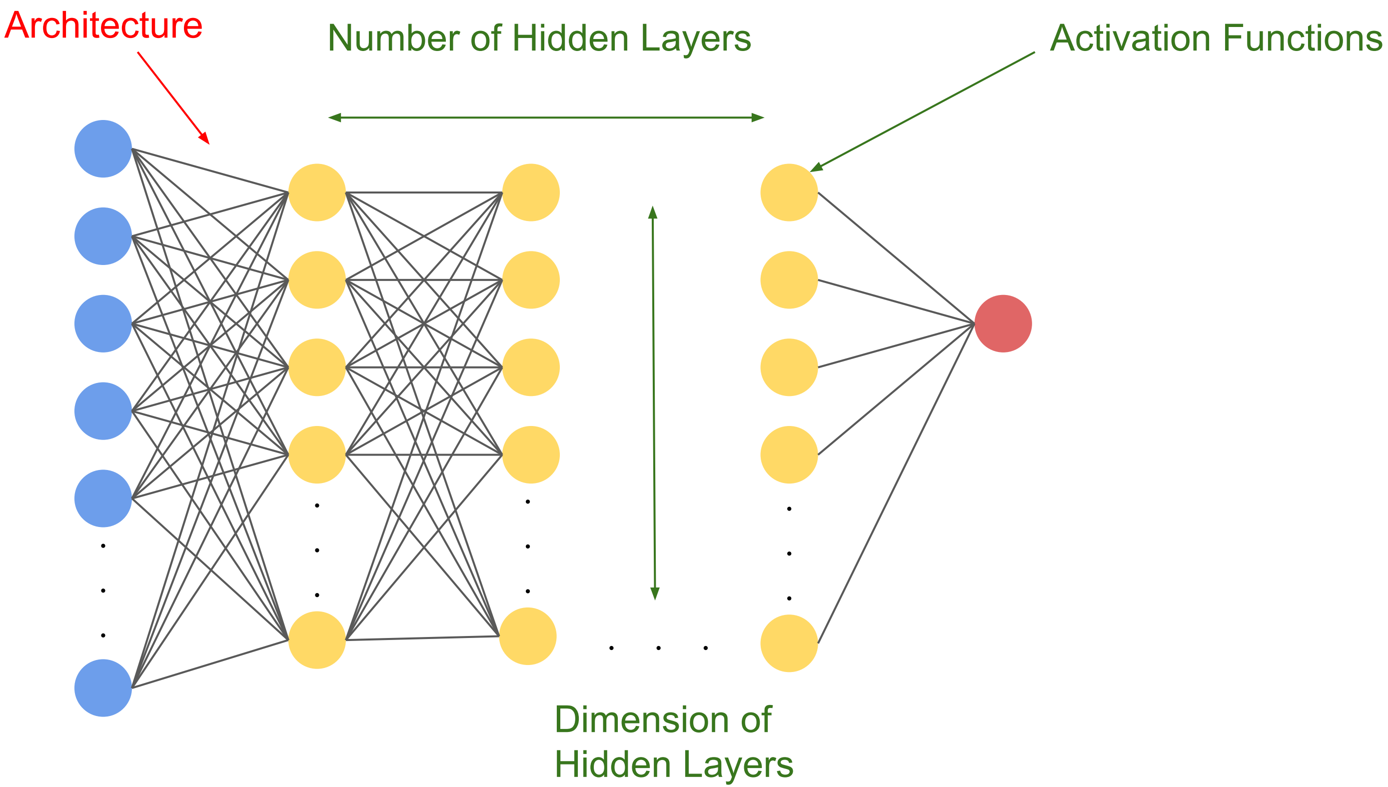


Knobs at our disposal - Depth vs. Width

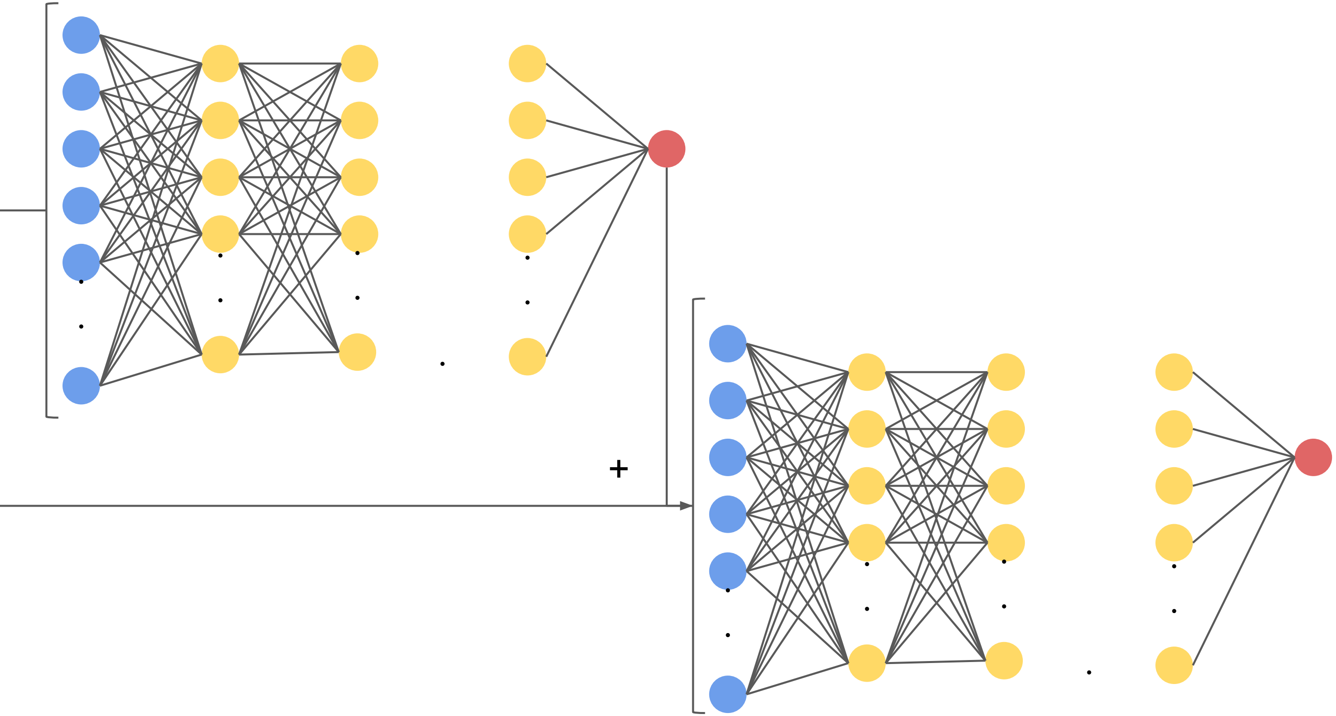


Goodfellow, 2014

Knobs at our disposal

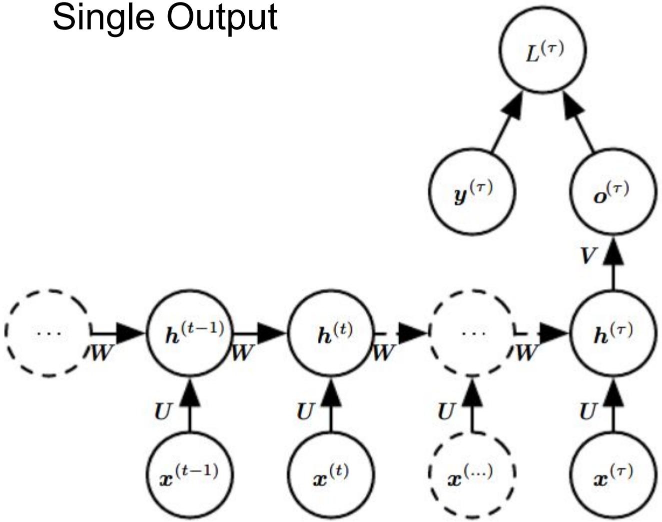


Architecture - Residual Networks

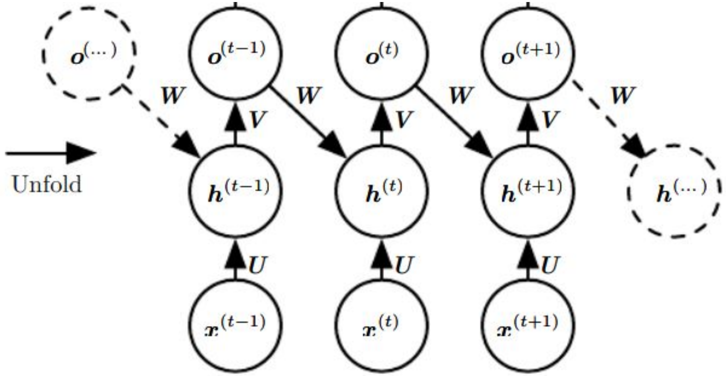


Architecture - Recurrent Neural Networks

Sequence with
Single Output

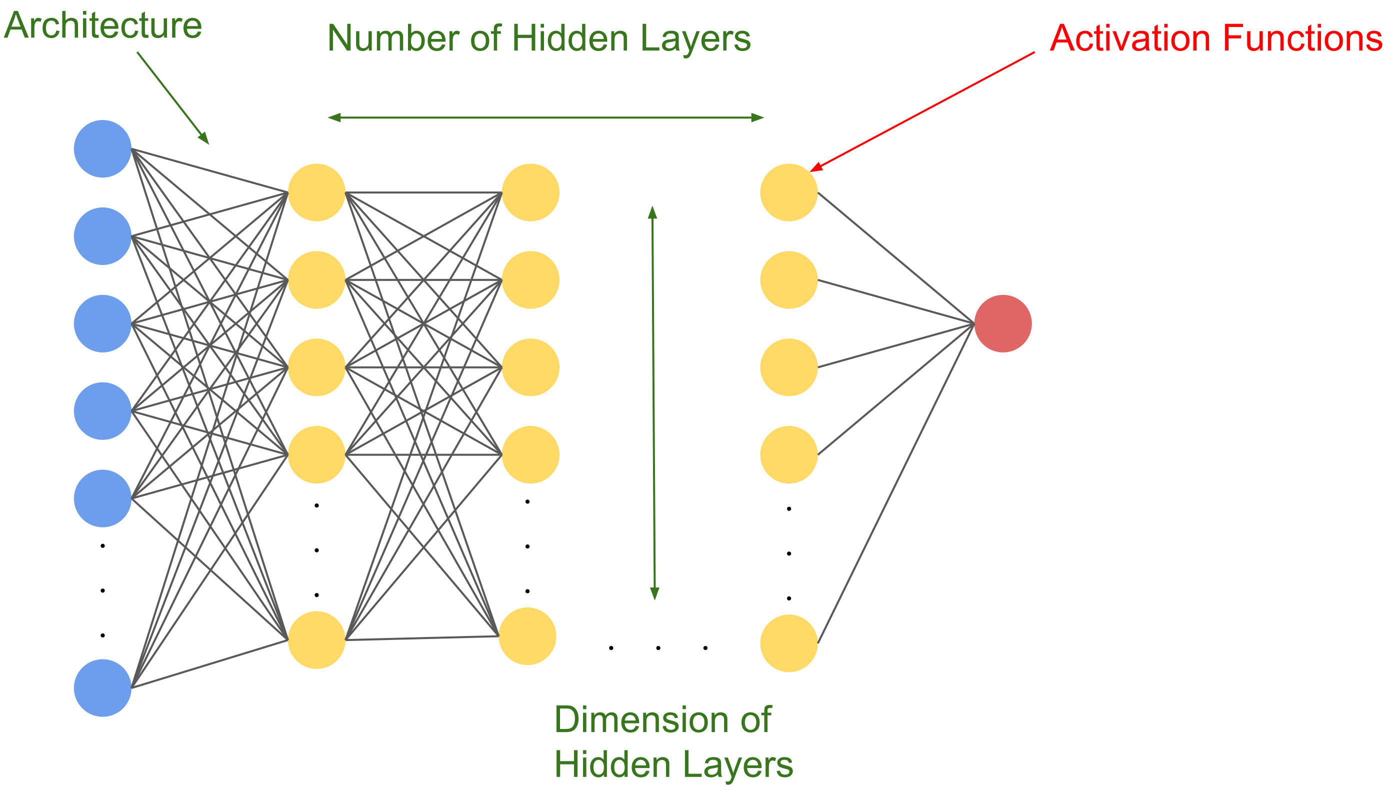


Sequence with
Output at each
Step

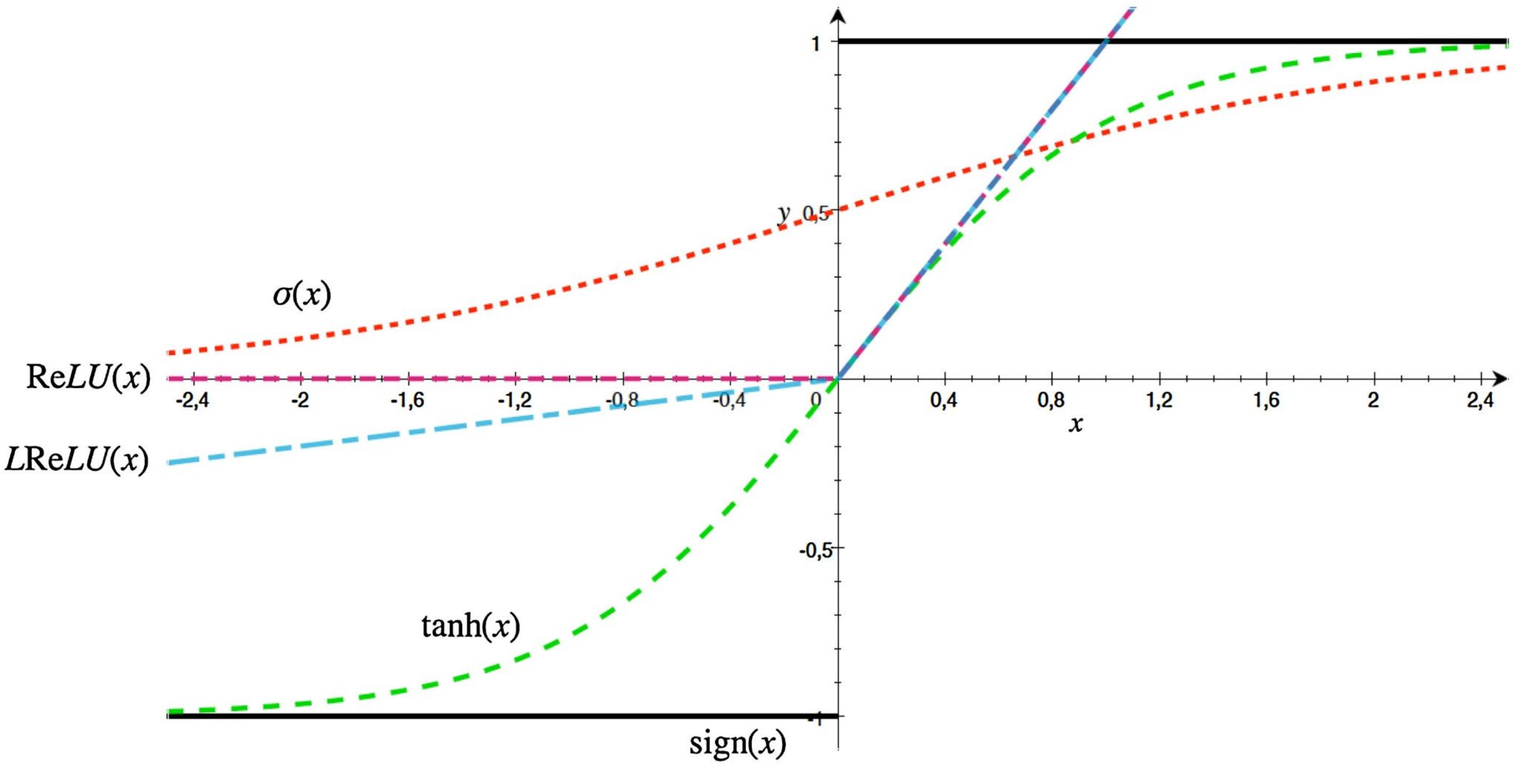


Goodfellow et. al, [Deep Learning]

Knobs at our disposal



Activation Functions



Maier 2019

Why Neural Nets?

Universal Approximation Theorem

🙌

Any feedforward network with a hidden layer and “squashing” activation can arbitrarily fit any function

🙌

Hornik, 1989

Universal Approximation Theorem

🙌

Any feedforward network with a hidden layer and “squashing” activation can arbitrarily fit any function

🙌

(provided there are enough hidden units)

[Hornik, 1989]

Also it just works really well

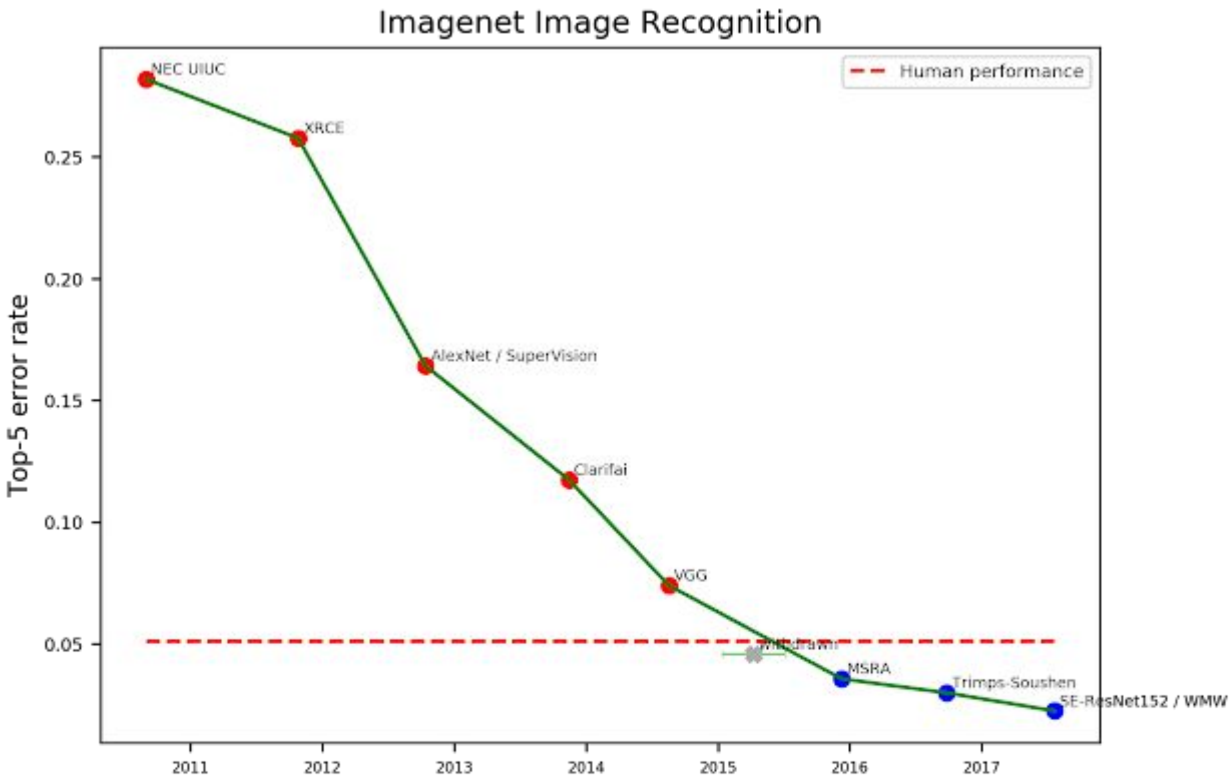


Image Classification

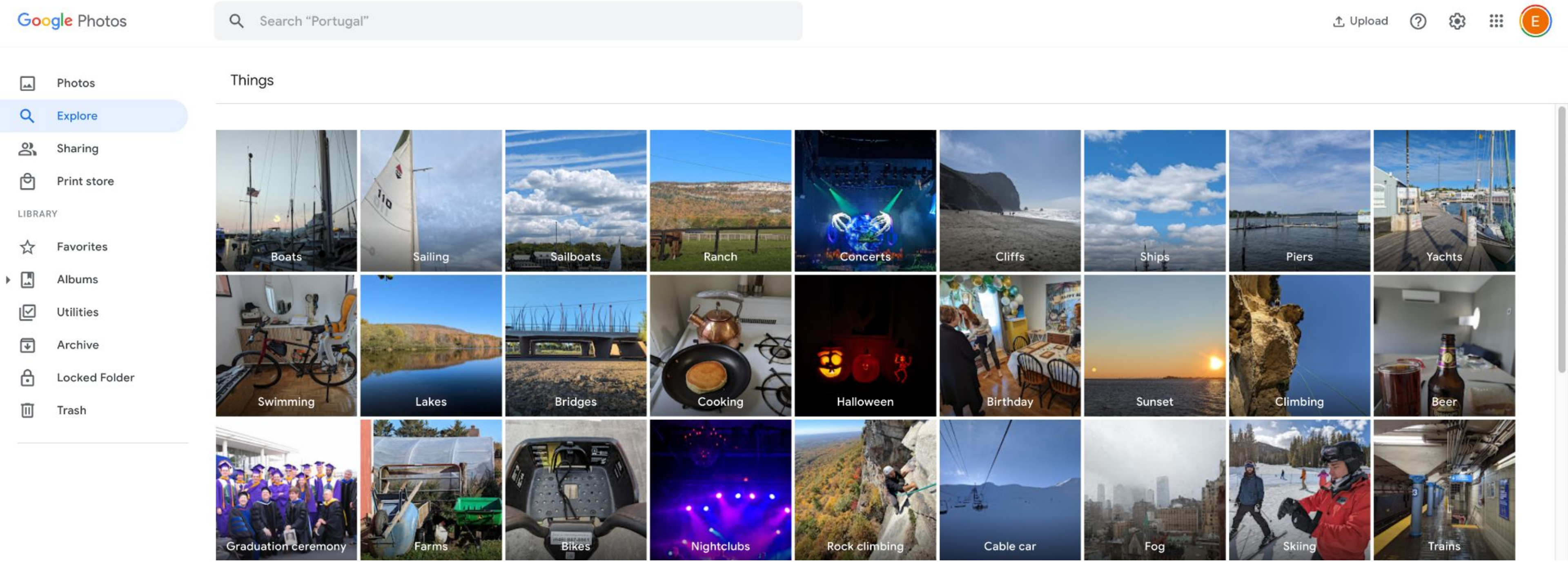
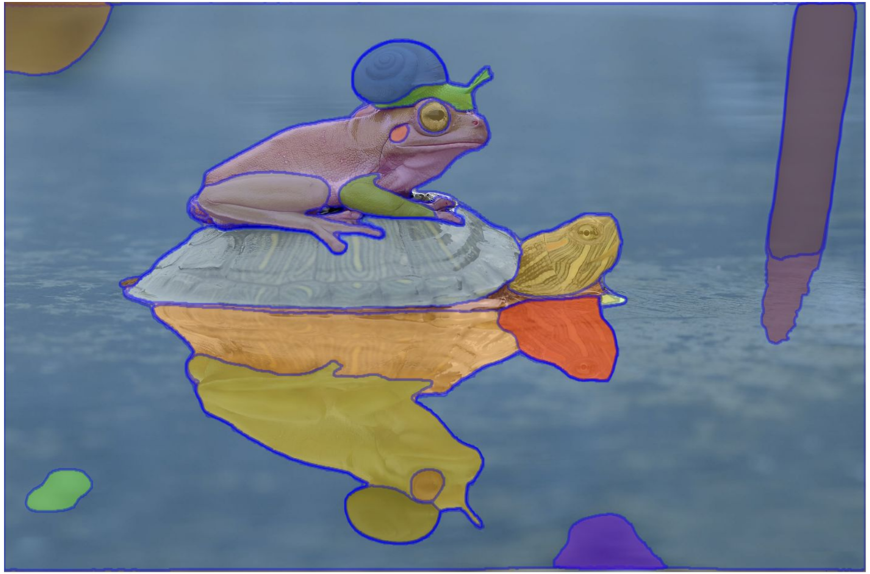


Image Segmentation



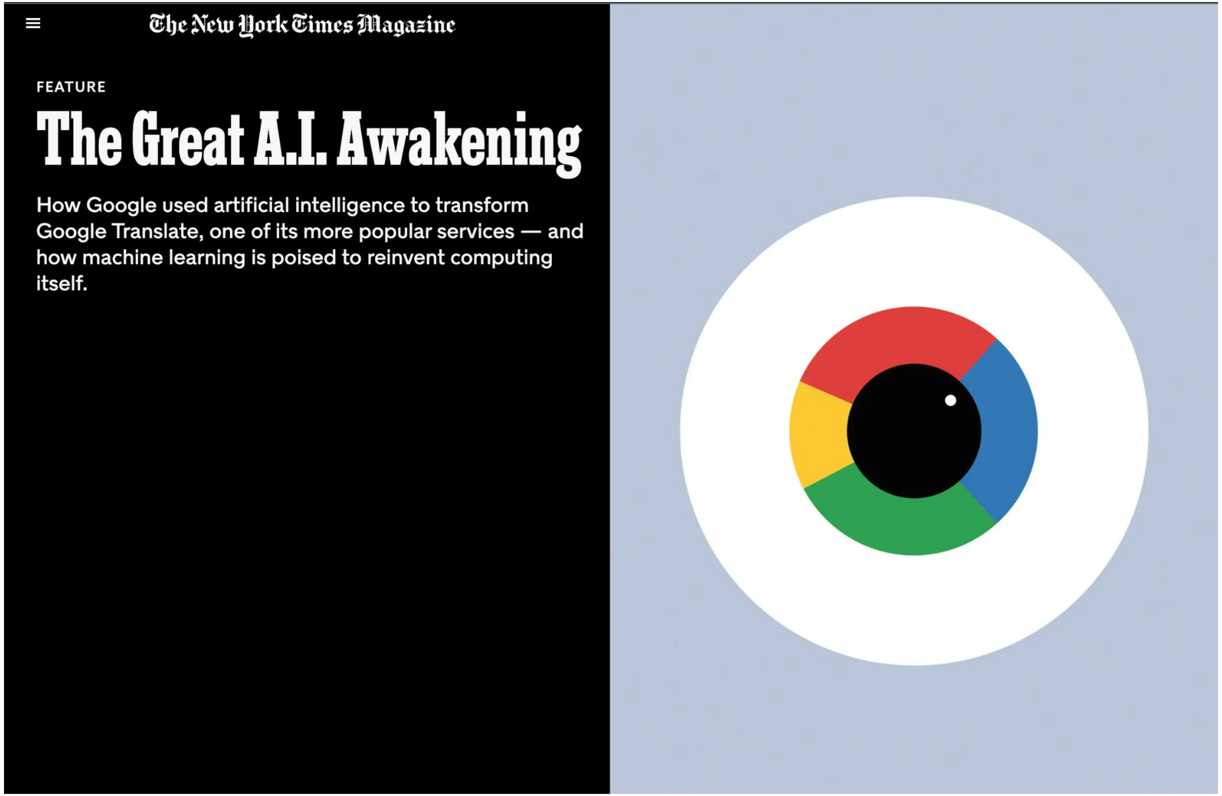
Segment Anything

Image Generation

"a photograph of an astronaut riding a horse"

Reference

Translation


The New York Times Magazine

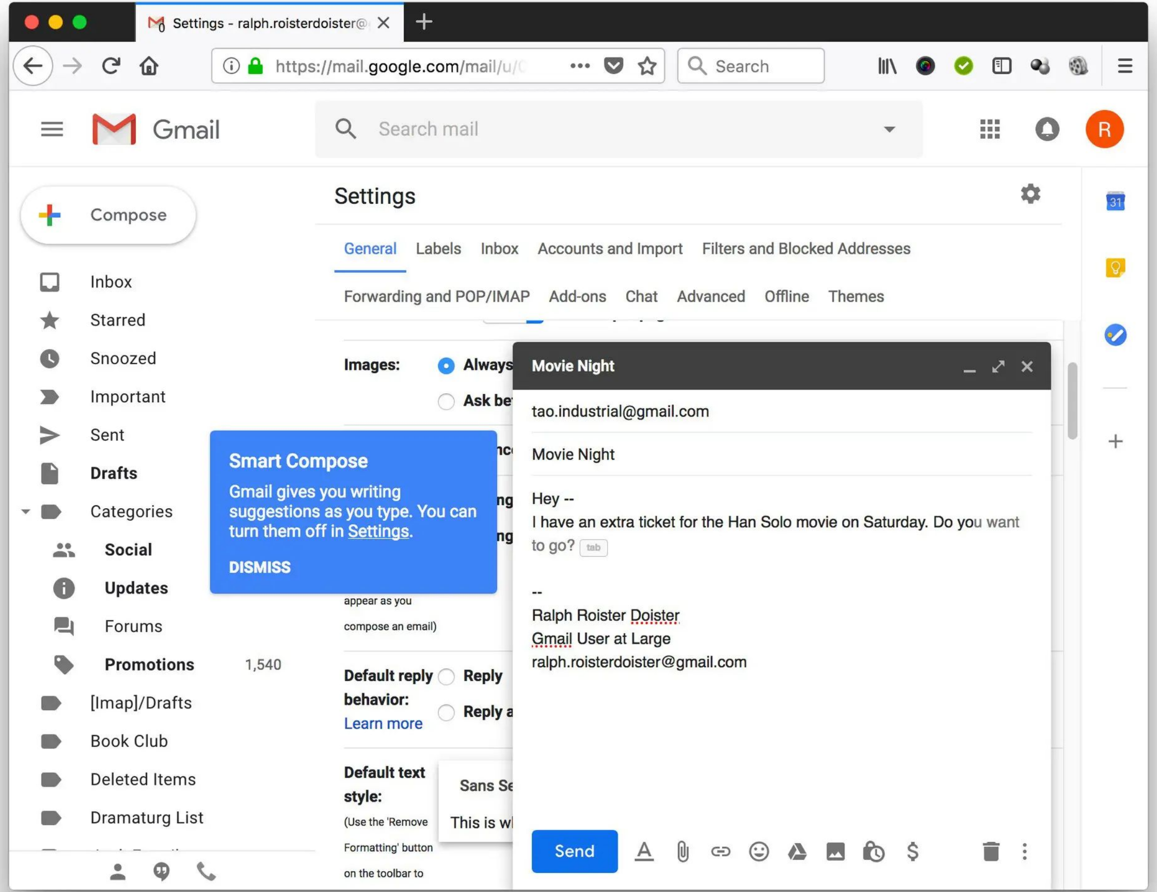
FEATURE

The Great A.I. Awakening

How Google used artificial intelligence to transform Google Translate, one of its more popular services — and how machine learning is poised to reinvent computing itself.

Link

Autocomplete



ChatGPT

Default (GPT-3.5)

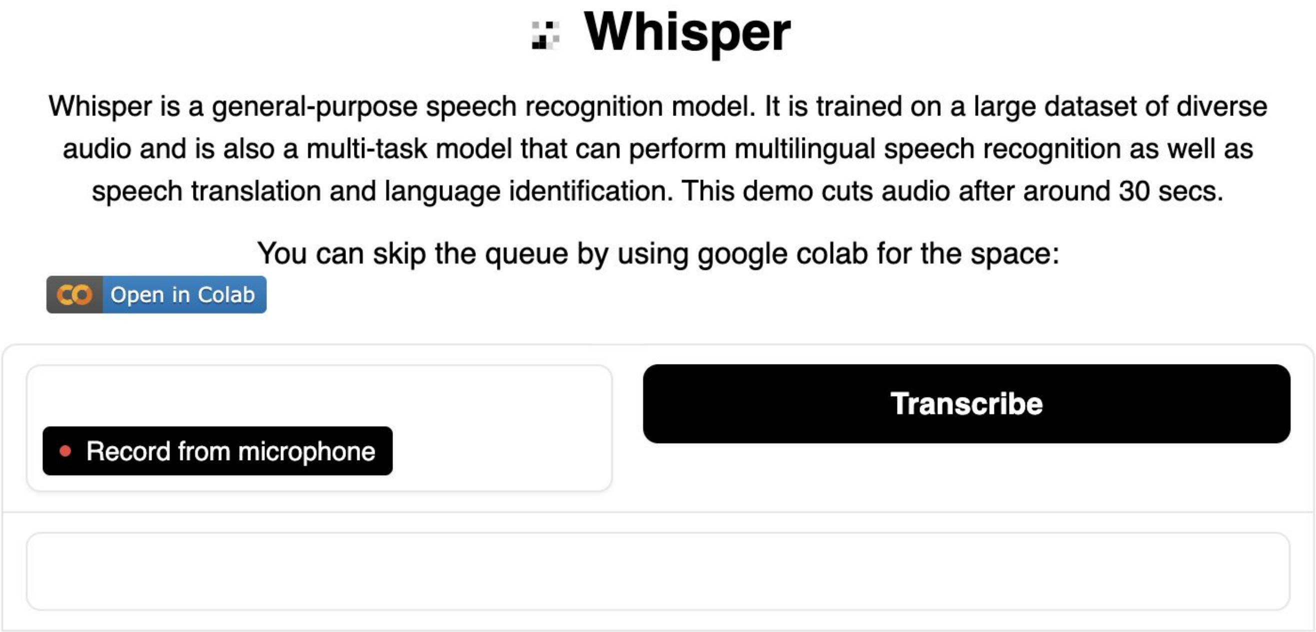
ER What breakthroughs has deep learning been responsible for over the last 10 years?

Deep learning has been responsible for numerous breakthroughs in various fields over the last decade. Here are some of the notable breakthroughs and advancements:

1. Image and Video Recognition:
   - Image classification and object recognition: Deep learning techniques, especially Convolutional Neural Networks (CNNs), have achieved remarkable accuracy in tasks like ImageNet classification.
   - Object detection: Models like YOLO (You Only Look Once) and Faster R-CNN have made significant progress in real-time object detection in images and videos.
   - Image and video segmentation: Deep learning models have improved the accuracy and speed of pixel-wise image and video segmentation, with applications in medical imaging, autonomous vehicles, and more.
2. Natural Language Processing (NLP):
   - Language models: Transformer-based models like GPT-3 and BERT have revolutionized NLP by achieving state-of-the-art results in various language understanding tasks.
   - Language generation: Deep learning models have been used for natural language generation tasks, including text summarization, chatbots, and content creation.
   - Multilingual NLP: Deep learning models have facilitated multilingual NLP applications, allowing for cross-lingual understanding and translation.
3. Reinforcement Learning:
   - AlphaGo: Deep reinforcement learning was at the core of DeepMind's AlphaGo, which defeated the world champion Go player. It marked a significant milestone in AI's ability to tackle complex board games.
4. Healthcare:
   - Medical image analysis: Deep learning has enabled better diagnosis and analysis of medical images, such as X-rays, CT scans, and MRIs, helping in early disease detection.
   - Drug discovery: Deep learning models have accelerated drug discovery by predicting potential drug candidates and their interactions.
5. Autonomous Vehicles:
   - Self-driving cars: Deep learning algorithms have played a crucial role in the development of autonomous vehicles, allowing them to perceive and navigate the environment.
6. Speech Recognition:
   - Automatic speech recognition (ASR): Deep learning has improved the accuracy of ASR systems, making voice assistants and transcription services more effective.

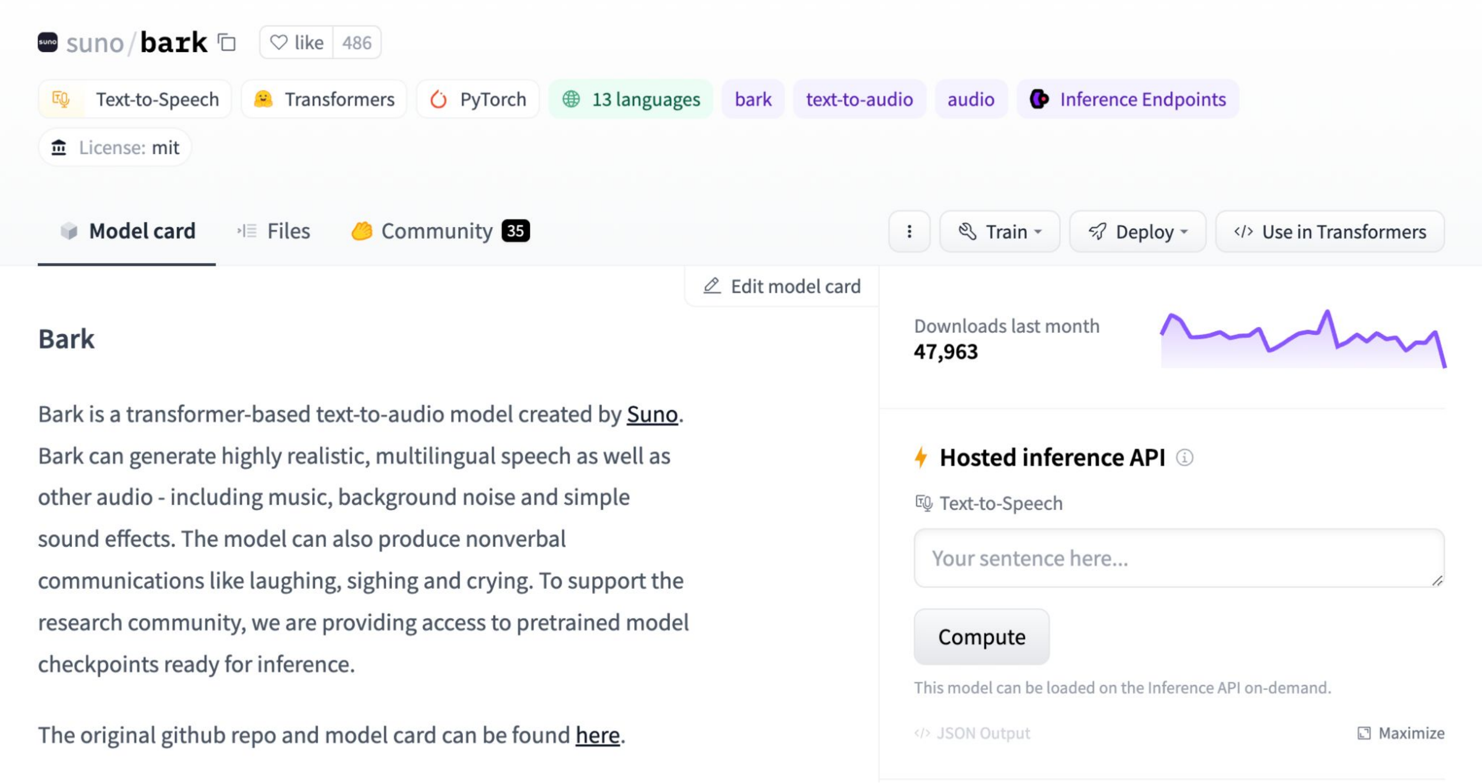
Speech to Text

https://huggingface.co/spaces/openai/whisper



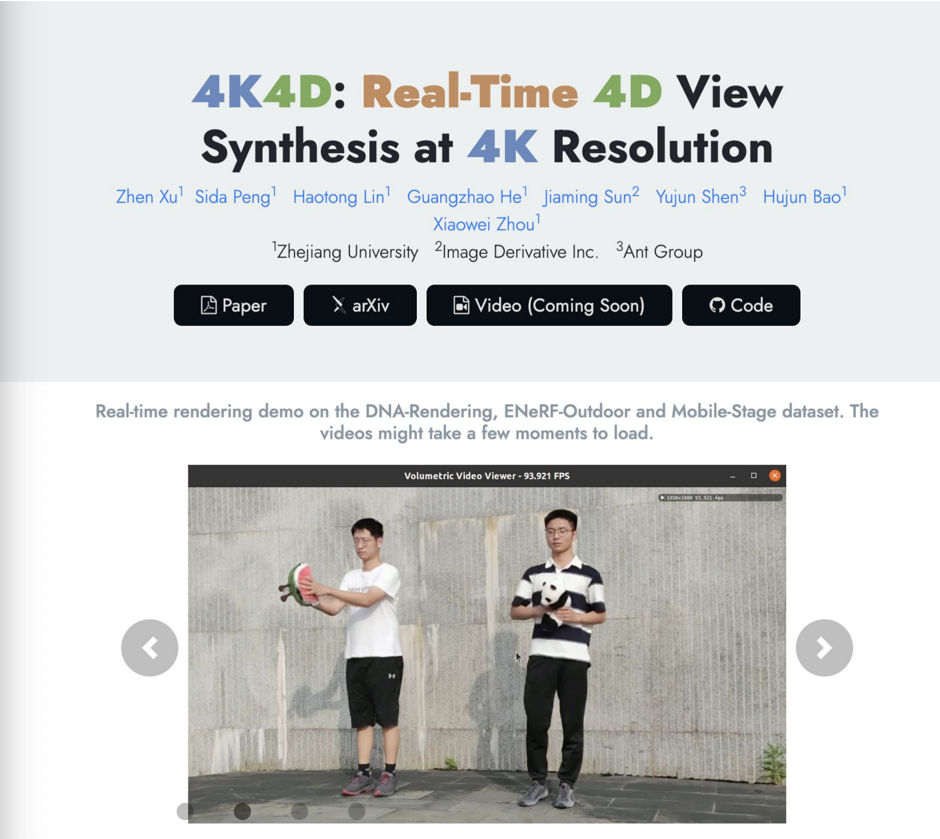
Text to Speech

https://huggingface.co/suno/bark



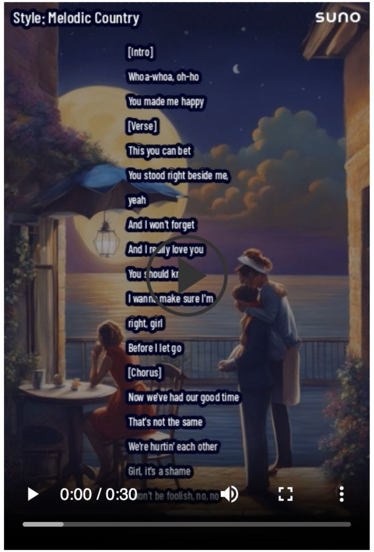
4D View Synthesis

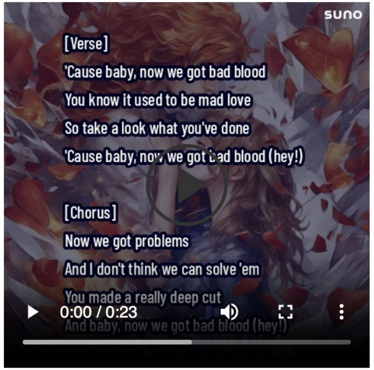
https://zju3dv.github.io/4k4d/

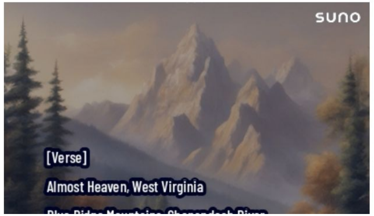


Text to Music

https://suno-ai.notion.site/Chirp-v1-Examples-cc71e6c0c79f4e03acf39aa5d5a3dd09







The Bitter Lesson

- Compute Rules Everything Around Me

Optimization

Recall Optimizing Linear Regression

$$\mathcal{L} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

$$\mathcal{L} = \frac{1}{n}\sum_{i=1}^{n}\left(y_i - \vec{X}_i \cdot \vec{\beta}\right)^2$$

$$\mathcal{L} = \frac{1}{n}\sum_{i=1}^{n}\left(y_i - \sum_{j=0}^{p} X_{ij}\beta_j\right)^2$$

$$\frac{\partial \mathcal{L}}{\partial \vec{\beta}} = 0$$

$$\vec{\beta} = \left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\vec{y}$$

Optimizing a Neural Network

$$\mathcal{L}\left(f(\mathbf{X};\theta);\mathbf{y}\right) = \frac{1}{n}\sum_{i=1}^{n}\left(y_i - f(\mathbf{X}_i;\theta)\right)^2$$

$$\frac{\partial \mathcal{L}}{\partial \theta_j} = 0$$

Stochastic Gradient Descent

1. Sample a Batch of data, m, and calculate the loss on this

$$\frac{1}{m}\sum_{i=1}^{m}\mathcal{L}\left(f(\mathbf{X}_i;\theta);\mathbf{y}_i\right)$$

$$\mathcal{L}\left(f(\mathbf{X};\theta);\mathbf{y}\right)=\frac{1}{n}\sum_{i=1}^{n}\left(y_i-f(\mathbf{X}_i;\theta)\right)^2$$

$$\frac{\partial\mathcal{L}}{\partial\theta_j}=0$$

Stochastic Gradient Descent

$$\mathcal{L}\left(f(\mathbf{X};\theta);\mathbf{y}\right) = \frac{1}{n}\sum_{i=1}^{n}\left(y_i - f(\mathbf{X}_i;\theta)\right)^2$$

$$\frac{\partial \mathcal{L}}{\partial \theta_j} = 0$$

1. Sample a Batch of data, m, and calculate the loss on this

$$\frac{1}{m}\sum_{i=1}^{m}\mathcal{L}\left(f(\mathbf{X}_i;\theta);\mathbf{y}_i\right)$$

2. Calculate the gradient of the loss with respect to a parameter

$$\hat{g}_j = \frac{1}{m}\nabla_{\theta_j}\sum_{i=1}^{m}\mathcal{L}\left(f(\mathbf{X}_i;\theta);\mathbf{y}_i\right)$$

Stochastic Gradient Descent

$$\mathcal{L}\left(f(\mathbf{X};\theta);\mathbf{y}\right) = \frac{1}{n}\sum_{i=1}^{n}\left(y_i - f(\mathbf{X}_i;\theta)\right)^2$$

$$\frac{\partial \mathcal{L}}{\partial \theta_j} = 0$$

1. Sample a Batch of data, m, and calculate the loss on this

$$\frac{1}{m}\sum_{i=1}^{m}\mathcal{L}\left(f(\mathbf{X}_i;\theta);\mathbf{y}_i\right)$$

2. Calculate the gradient of the loss with respect to a parameter

$$\hat{g}_j = \frac{1}{m}\nabla_{\theta_j}\sum_{i=1}^{m}\mathcal{L}\left(f(\mathbf{X}_i;\theta);\mathbf{y}_i\right)$$

3. Update the parameter proportional to the gradient.

$$\theta_j \leftarrow \theta - \epsilon\hat{g}_j$$

Stochastic Gradient Descent (SGD)

- We don’t have to hold all of the data in memory.
- We don’t have to perform calculations on all of the data at the same time.
- We don’t have to calculate all of the gradients at the same time.
- Surprisingly, this noisy approximation works!

Bottou 1998

Stochastic Gradient Descent (SGD)

$$\theta_j \leftarrow \theta - \epsilon \hat{g}_j$$

- There are many modifications to this equation.
- The learning rate, ϵ, is a crucial hyperparameter.
- The learning rate can change as optimization progresses (e.g. decay).
- We can scale parameter-specific learning rates based upon their gradients (e.g. Adagrad).
- We can accelerate learning ("momentum") by using the gradient from the previous iteration.

Bottou 1998

Backprop

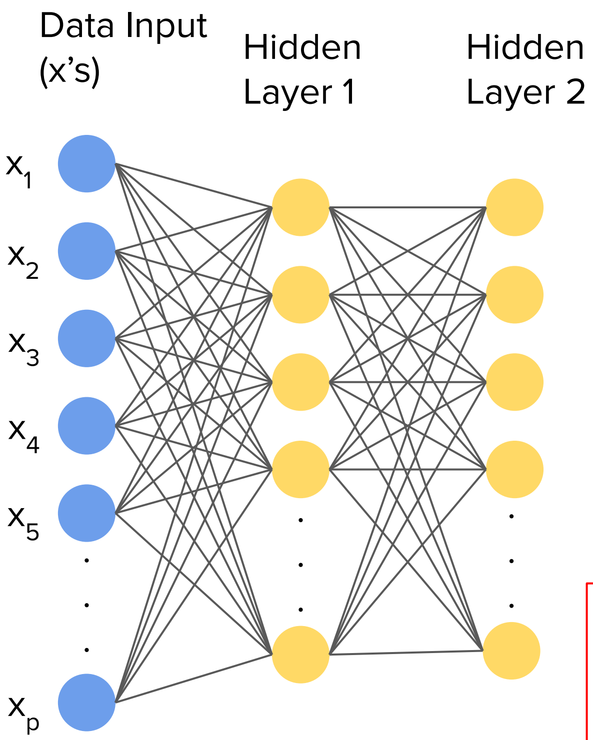
The Chain Rule of Calculus

$$y = g(x)$$
$$z = f(g(x)) = f(y)$$

$$\frac{dz}{dx} = \frac{dz}{dy}\frac{dy}{dx}$$

$$\frac{\partial z}{\partial x_i} = \sum_j \frac{\partial z}{\partial y_j}\frac{\partial y_j}{\partial x_i}$$

Feed Forward Network / Multilayer Perceptron



$$h^{(2)}_1 = f(b^{(2)}_1 + m^{(2)}_{11} * \mathbf{h^{(1)}_1} + m^{(2)}_{12} * \mathbf{h^{(1)}_2} + \dots m^{(2)}_{1L(1)} * \mathbf{h^{(1)}_{L(1)}})$$

$$= f(b^{(2)}_1 +$$
$$m^{(2)}_{11} * f(b^{(1)}_1 + m^{(1)}_{11} * x_1 + m^{(1)}_{12} * x_2 + \dots m^{(1)}_{1p} * x_p)$$
$$+ m^{(2)}_{12} * f(b^{(1)}_2 + m^{(1)}_{21} * x_1 + m^{(1)}_{22} * x_2 + \dots m^{(1)}_{2p} * x_p)$$
$$+ \dots$$
$$m^{(2)}_{1L(1)} * f(b^{(1)}_{L(1)} + m^{(1)}_{L(1)1} * x_1 + m^{(1)}_{L(1)2} * x_2 + \dots m^{(1)}_{L(1)p} * x_p)$$
$$)$$

$$h^{(2)}_1 = f(b^{(2)}_1; m_1^{(2)}; h^{(1)}) \quad , \quad h^{(1)}_1 = f(b^{(1)}_1; m_1^{(1)}; x)$$
$$h^{(2)}_1 = f(b^{(2)}_1; m_1^{(2)}; f(b^{(1)}_1; m_1^{(1)}; x))$$

Chain Rule ➜ Backprop

$$y = g(x)$$
$$z = f(g(x)) = f(y)$$

$h^{(2)}_1 = f(b^{(2)}_1; m_1^{(2)}; h^{(1)}_1)$, $h^{(1)}_1 = f(b^{(1)}_1; m_1^{(1)}; x)$

$h^{(2)}_1 = f(b^{(2)}_1; m_1^{(2)}; f(b^{(1)}_1; m_1^{(1)}; x))$

$$\frac{dz}{dx} = \frac{dz}{dy}\frac{dy}{dx}$$

$$\frac{\partial h_1^{(2)}}{\partial m_1^{(1)}} = \sum_j \frac{\partial h_1^{(2)}}{\partial h_j^{(1)}} \frac{\partial h_j^{(1)}}{\partial m_1^{(1)}}$$

$$\frac{\partial z}{\partial x_i} = \sum_j \frac{\partial z}{\partial y_j}\frac{\partial y_j}{\partial x_i}$$

Chain Rule ➜ Backprop

$$y = g(x)$$
$$z = f(g(x)) = f(y)$$

$h^{(2)}_1 = f(b^{(2)}_1;m_1^{(2)};h^{(1)}_1)$, $h^{(1)}_1 = f(b^{(1)}_1;m_1^{(1)};x)$

$h^{(2)}_1 = f(b^{(2)}_1;m_1^{(2)};f(b^{(1)}_1;m_1^{(1)};x))$

$$\frac{dz}{dx} = \frac{dz}{dy}\frac{dy}{dx}$$

$$\frac{\partial h_1^{(2)}}{\partial m_1^{(1)}} = \sum_j \frac{\partial h_1^{(2)}}{\partial h_j^{(1)}} \frac{\partial h_j^{(1)}}{\partial m_1^{(1)}}$$

$$\frac{\partial z}{\partial x_i} = \sum_j \frac{\partial z}{\partial y_j}\frac{\partial y_j}{\partial x_i}$$

Backpropagation

We can efficiently compute gradients for all parameters by:

1. Computing in reverse order.
2. Storing and updating gradient computations.