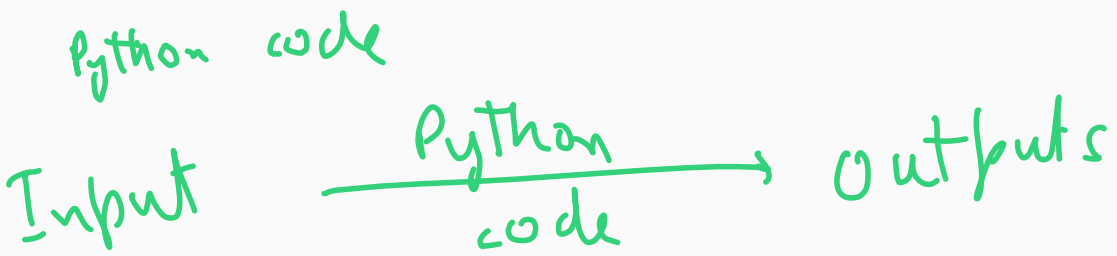
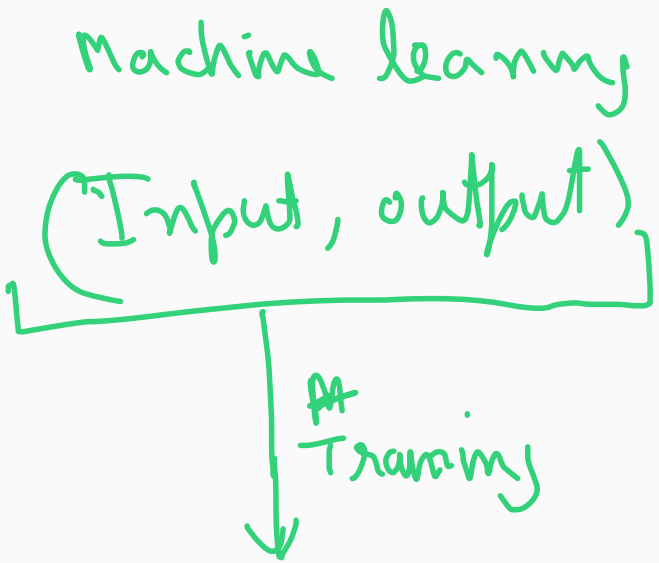
Neural Networks: Using pre-trained models

Vikas Dhiman

January 30, 2024

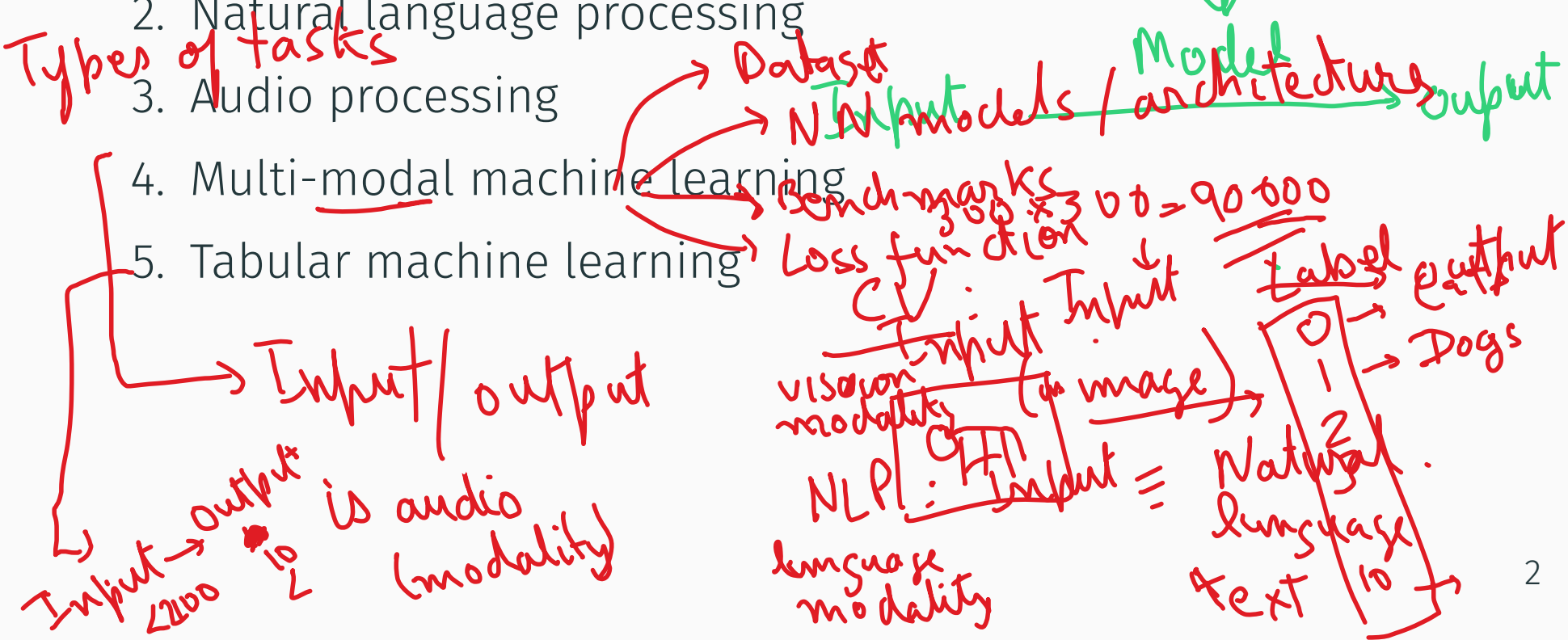
Machine learning problems





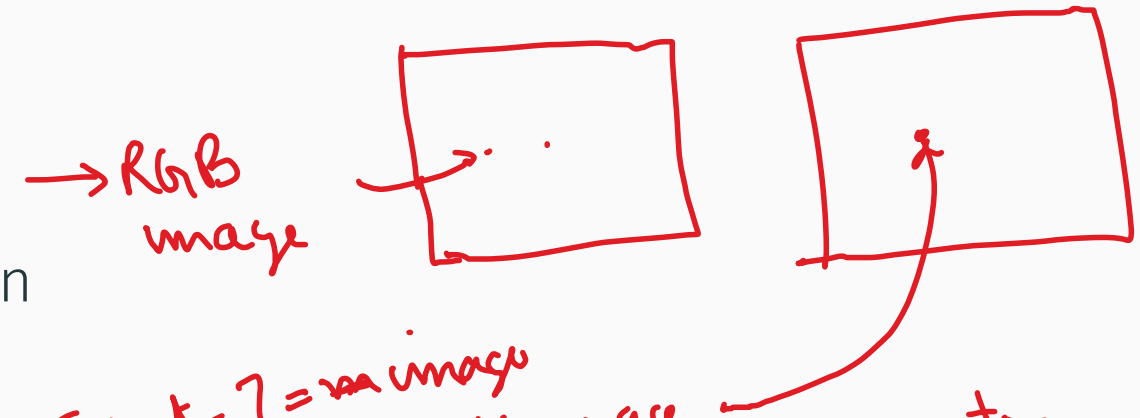
Classified on basis of type of input

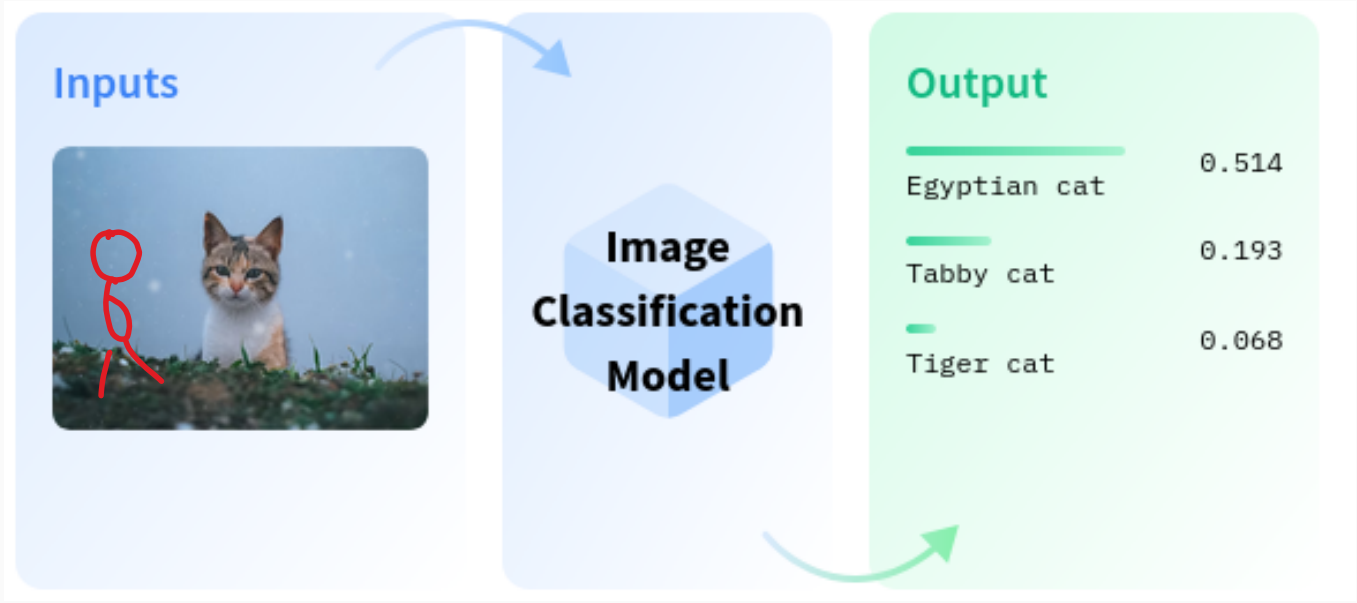
1. Computer Vision
2. Natural language processing
3. Audio processing
4. Multi-modal machine learning
5. Tabular machine learning



Computer vision

1. Image Classification
2. Object Detection
3. Depth Estimation
4. Image Segmentation
5. Image-to-Image (example drawing to realistic picture)
6. Mask Generation
7. Video Classification





. Input = image

Output =

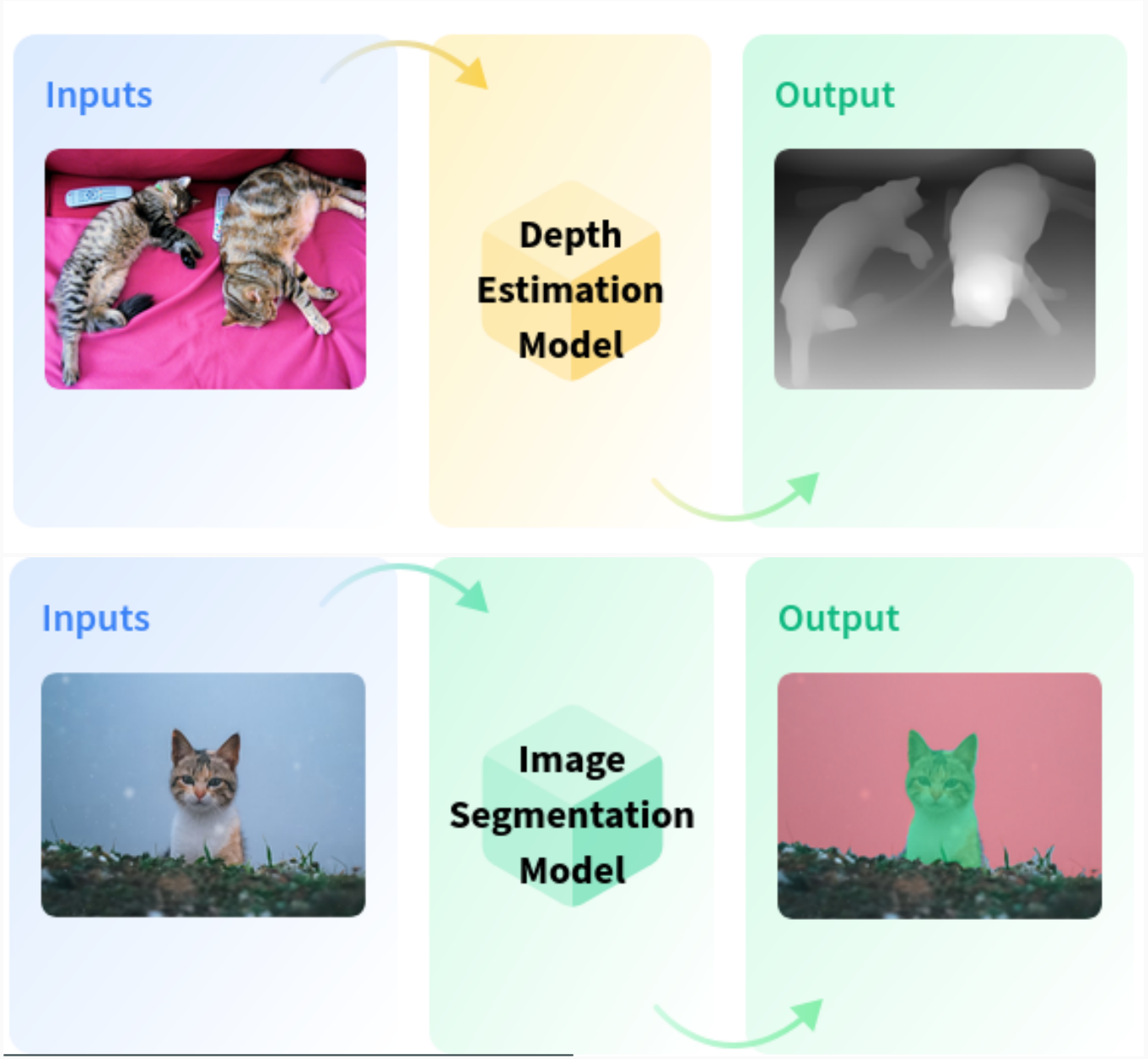
[Bounding box (location, size, label)

]

[1]Image source: huggingface.co

Hugging face tasks

- Image classification pre-trained Colab
- Object classification pre-trained Colab

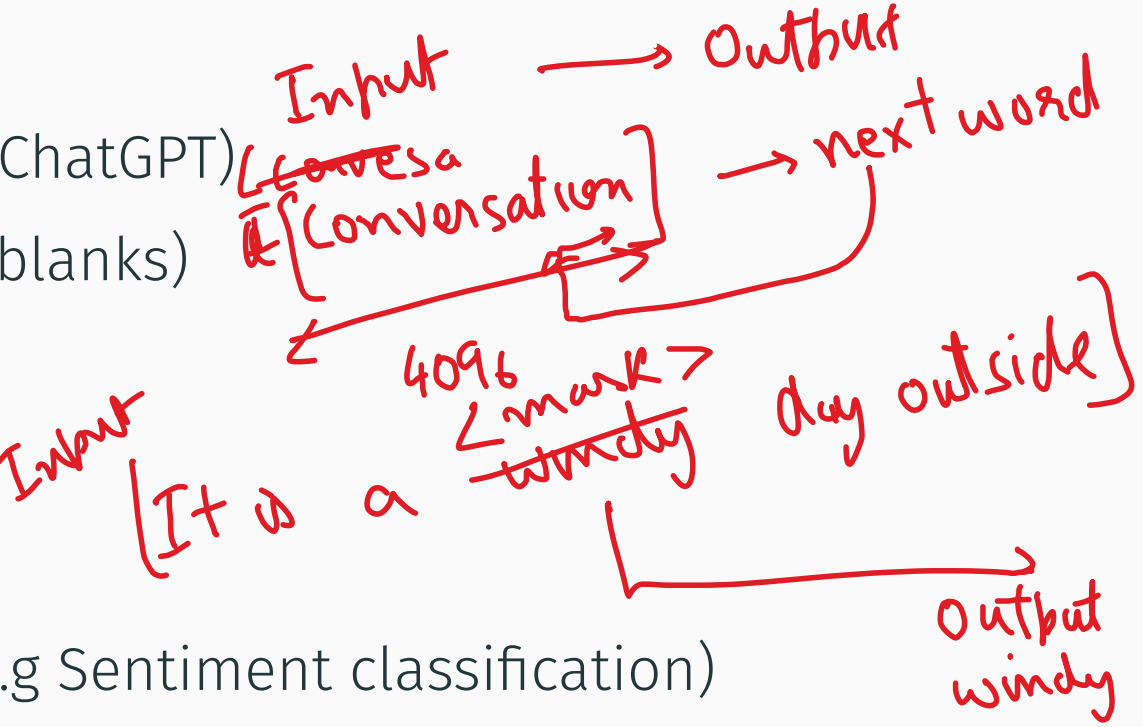


[2]

[2]Image source: huggingface.co

Natural Language Processing

1. Conversational (e.g. ChatGPT)
2. Fill-Mask (Fill in the blanks)
3. Question Answering
4. Sentence Similarity
5. Summarization
6. Text Classification (e.g Sentiment classification)
7. Text Generation (e.g. auto-completion)
8. Token Classification (e.g. noun, adjectives or person, place etc)
9. Translation

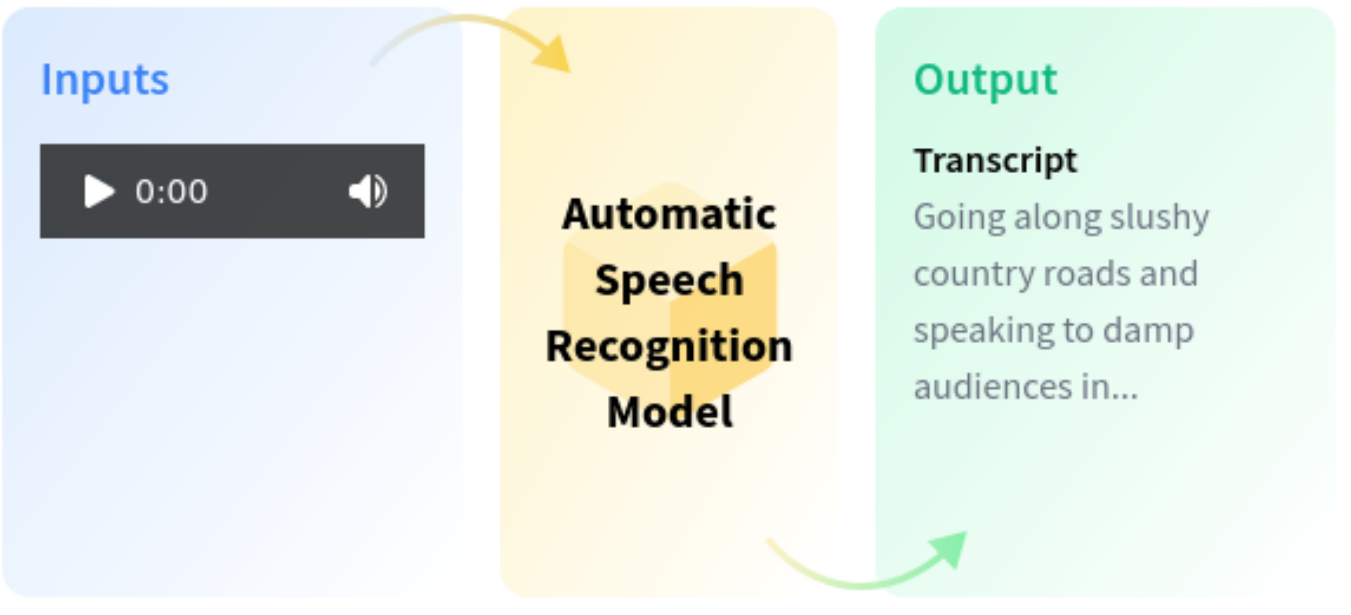


Can we run chatbots

- MetaAi LLAMA Clib/Llama-2-13B-chat-GGML

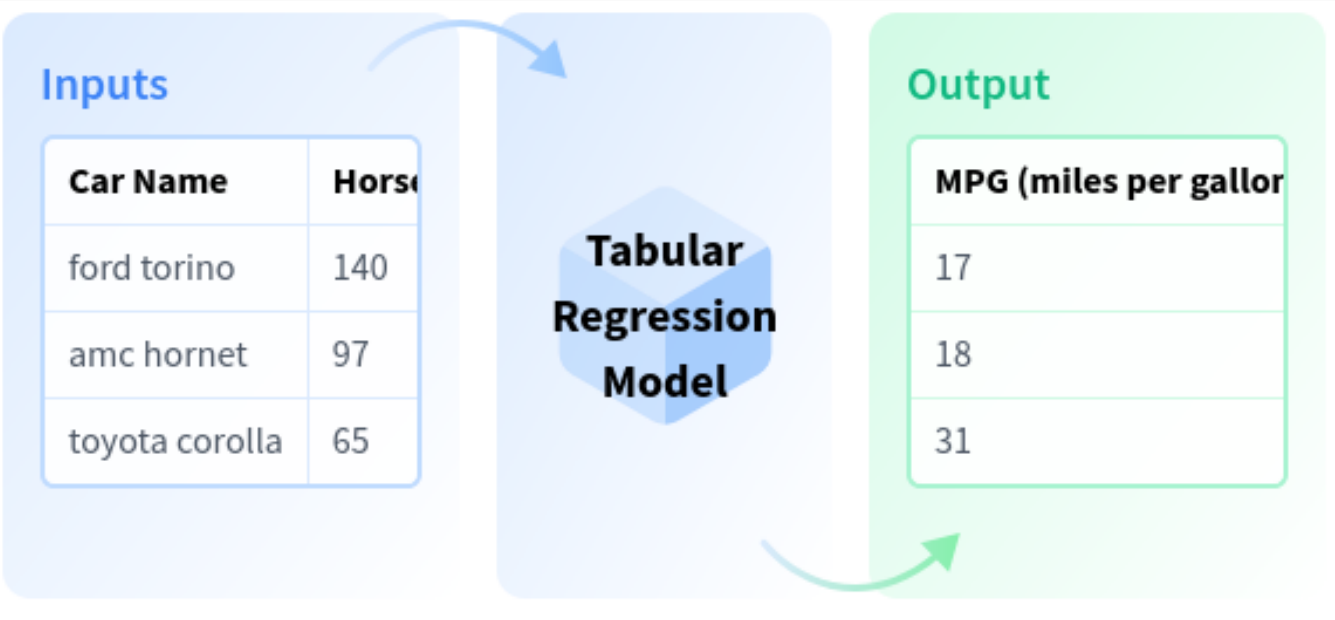
Audio

1. Audio Classification
2. Audio-to-Audio
3. Automatic Speech Recognition
4. Text-to-Speech



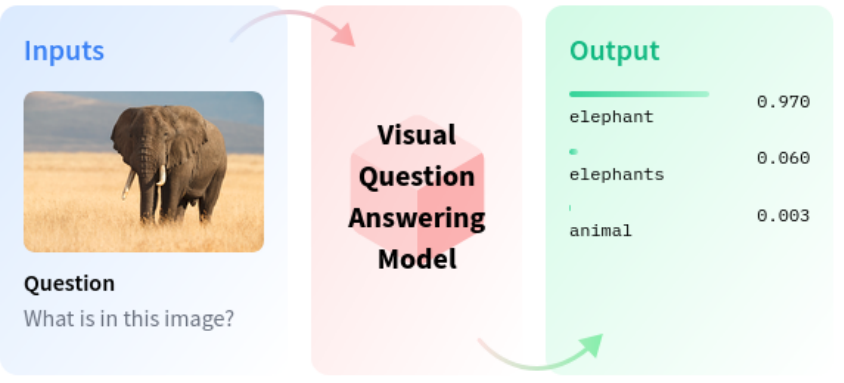
Tabular

1. Tabular Classification
2. Tabular Regression



Mutimodal

1. Document Question Answering
2. Feature Extraction
3. Image-to-Text
4. Text-to-Image
5. Text-to-Video
6. Visual Question Answering
7. Text-to-3D
8. Image-to-3D



Using pre-trained models

1. Same as using other people's code.
2. Have to find the model that has been trained on a "similar problem"
3. Options:
    - Search on Google/Google Scholar/Github (most options, least standardized)
    - Search on Tensorflow Hub: tensorflow.org/hub
    - Search on Pytorch Hub: pytorch.org/hub
    - Search on ONNX Hub: onnx.ai
    - Search on Huggingface tasks (fewest options, most standardized)

Homework 2: Using Pre-trained model

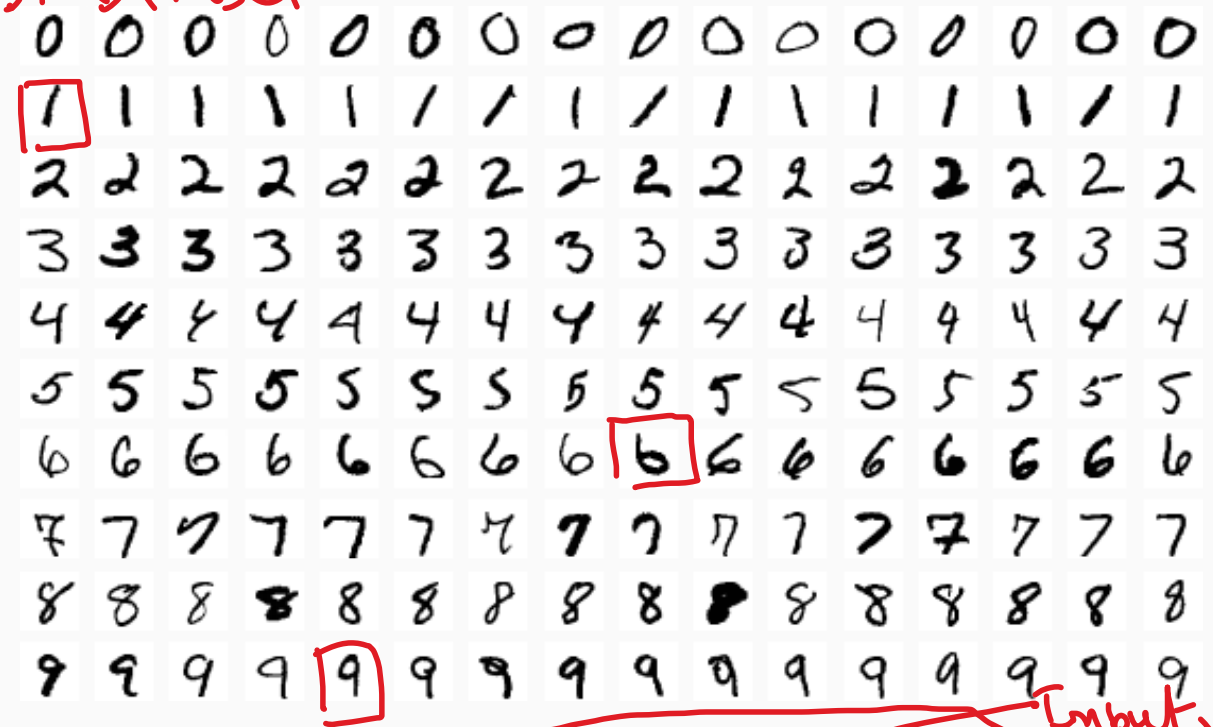
- Think of a project that you might want to do in this class.
- Find out the closest Hugging face task to your project
- Demonstrate that you can run at least one pre-trained Hugging face model on the standard task and a standard dataset on Google Colab or locally.

Dataset, Pre-processing, Models, and Learning

Data as Vectors: Pre-processing

Vectors = ↔ direction + magnitude ⟺ Scalars = 1D vectors

MNIST Dataset



Vectors = $\begin{bmatrix} 2. \\ 3. \\ 4. \\ 3.5 \\ 2.5 \end{bmatrix}$

$$\mathcal{D} = \{(x_1, y_1), \ldots, (x_i, y_i), \ldots, (x_n, y_n)\}$$

Input vector

output scalar

Input 28×28

labels

Outputs y

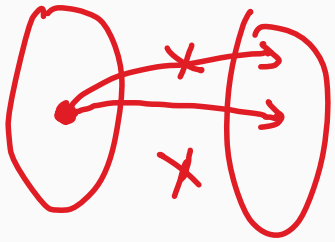
Dataset

90,000

= vector

x

Models as functions



A predictor as a function, $f : \mathbb{R}^d \mapsto \mathbb{R}$

1. Example: Linear Model: $f(\mathbf{x}; \mathcal{W}) = \mathbf{w}^T\mathbf{x} + w_0$
2. Example: Non-linear model (Two layer neural network) $f(\mathbf{x}; \mathcal{W}) = \mathbf{w}_2^\top \sigma(\mathbf{W}_1\mathbf{x} + \mathbf{w}_0)$, where $\sigma : \mathbb{R} \mapsto \mathbb{R}$ is some non-linear activation function like ReLU, sigmoid or **tanh**.

Loss functions and Learning

$f(x) = Linear(Act(Linear(x)))$

Two-layer NN

$X = Linear(Linear(x)) = Linear(x)$

$$R_{emp}(f, \mathbf{X}, \mathbf{y}) = \frac{1}{n}\sum_{i=1}^{n} l(y_i, \hat{y}_i)$$

Avg loss over entire dataset is called Empirical Risk

, where $\hat{y}_i = f(\mathbf{x}_i; \mathcal{W})$.

$R_{emp}(f, \mathbf{X}, \mathbf{y})$ is called the empirical risk.

Loss function = error function

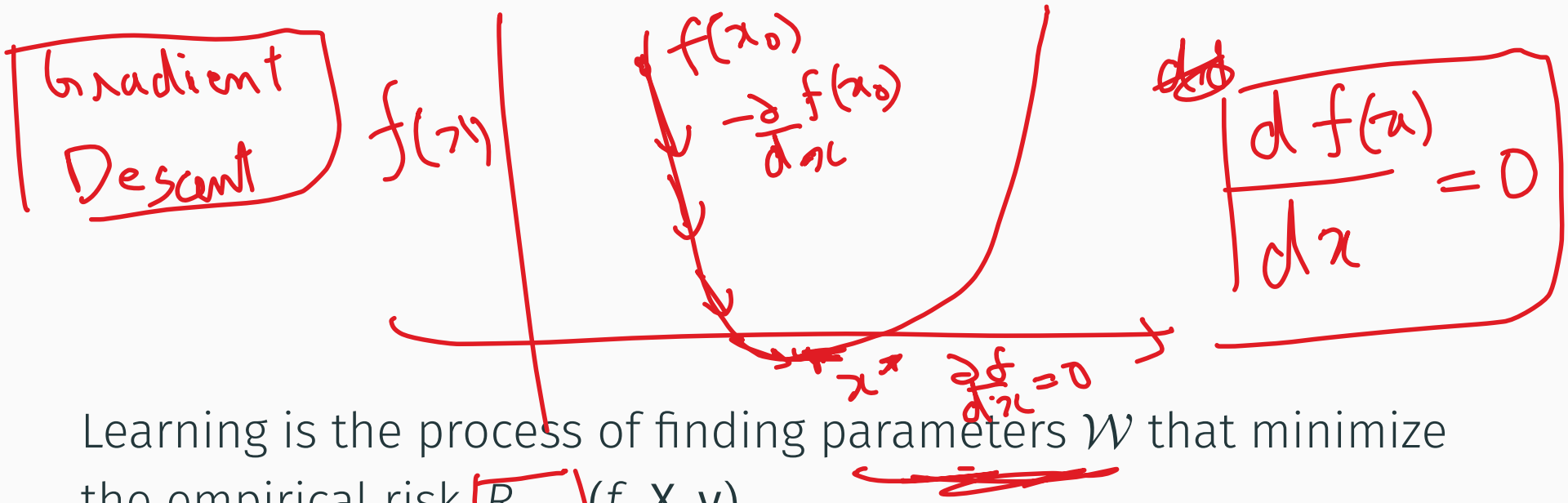
$\underline{x} \longrightarrow 9 = y$

9

$f(\underline{x}) = 8.5$ $\quad \hat{y}$

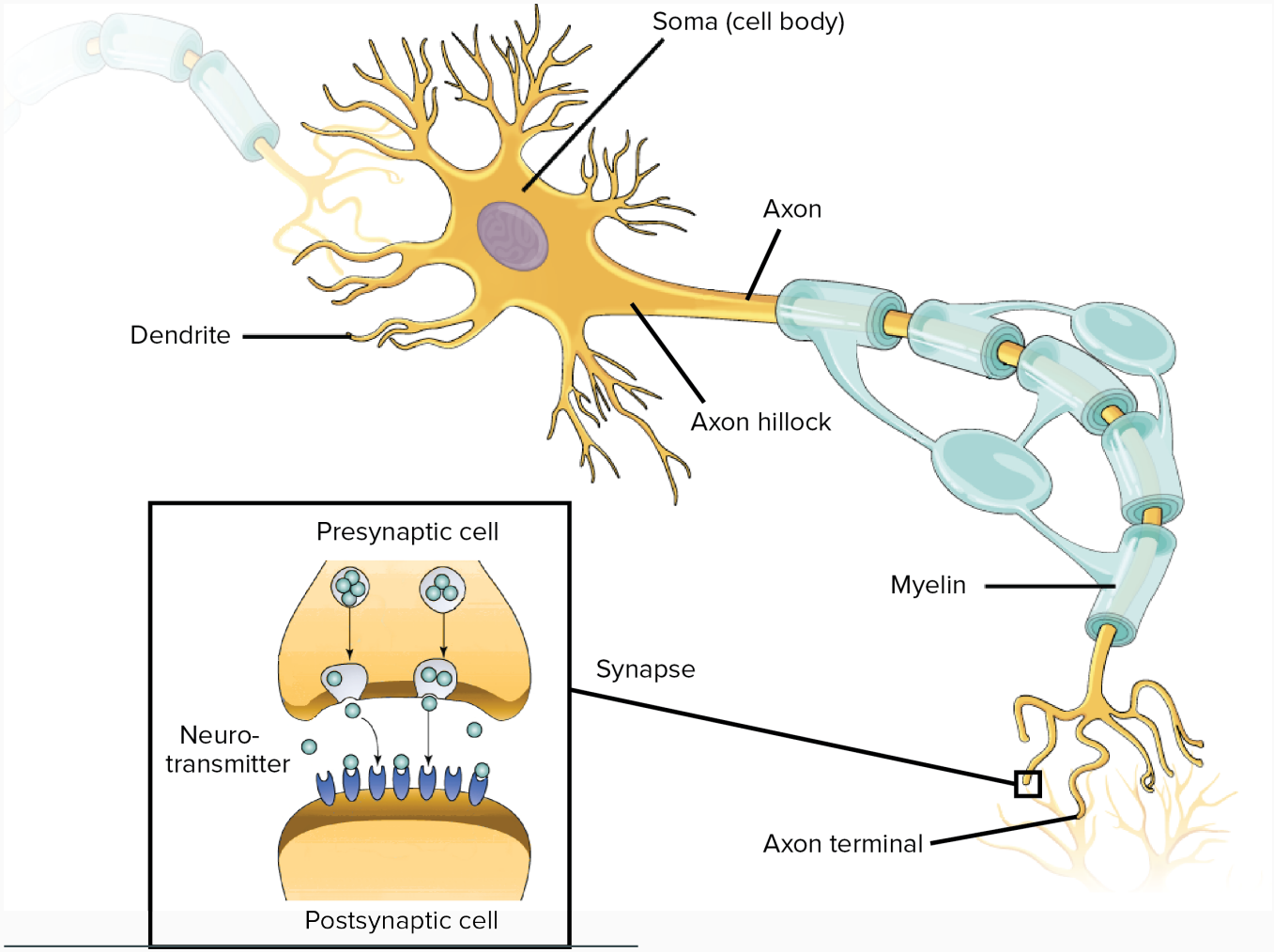
$9 - 8.5 = \|y - f(\underline{x})\|^2 = \text{Loss}$

Learning



Learning is the process of finding parameters $\mathcal{W}$ that minimize the empirical risk, $R_{\text{emp}}(f, \mathsf{X}, \mathsf{y})$.

Neural Networks: Biology vs Artificial

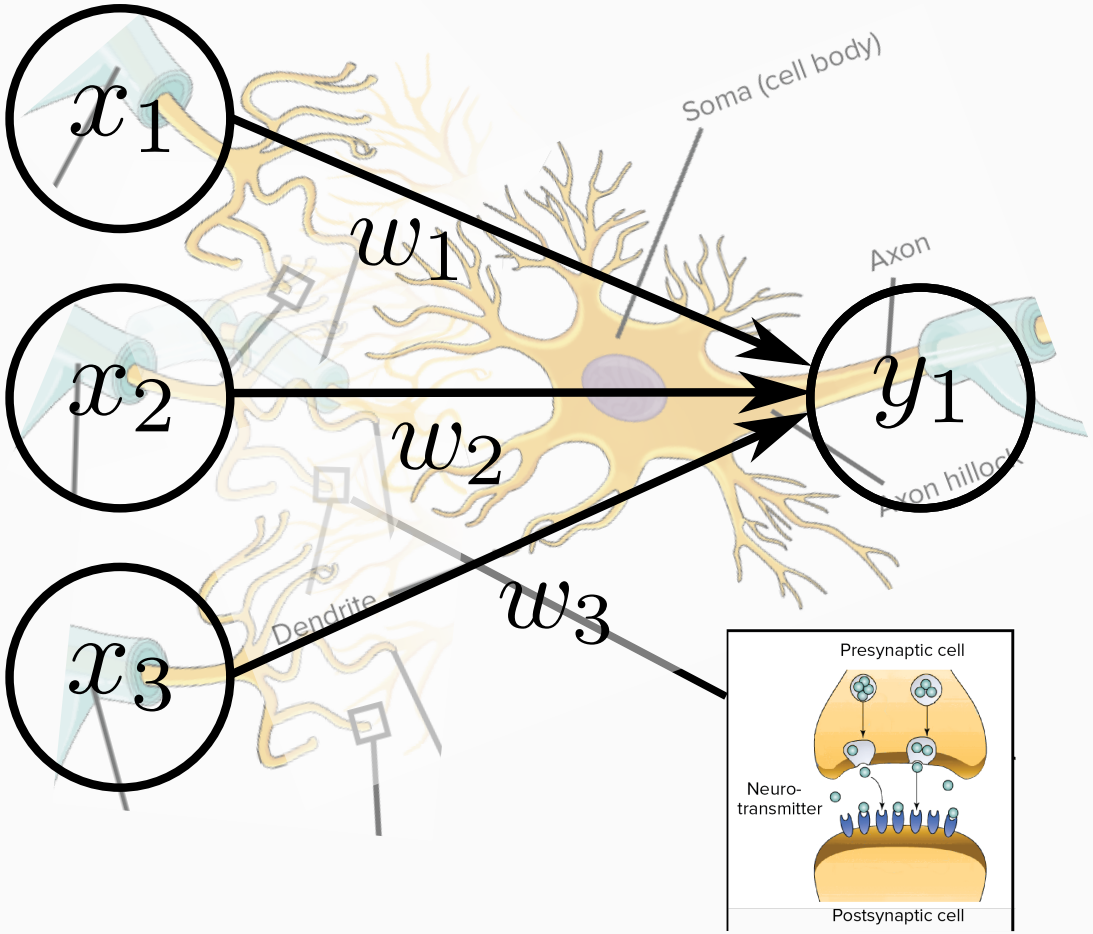


[3]

[3] Source: https://openstax.org/books/biology/pages/35-2-how-neurons-communicate

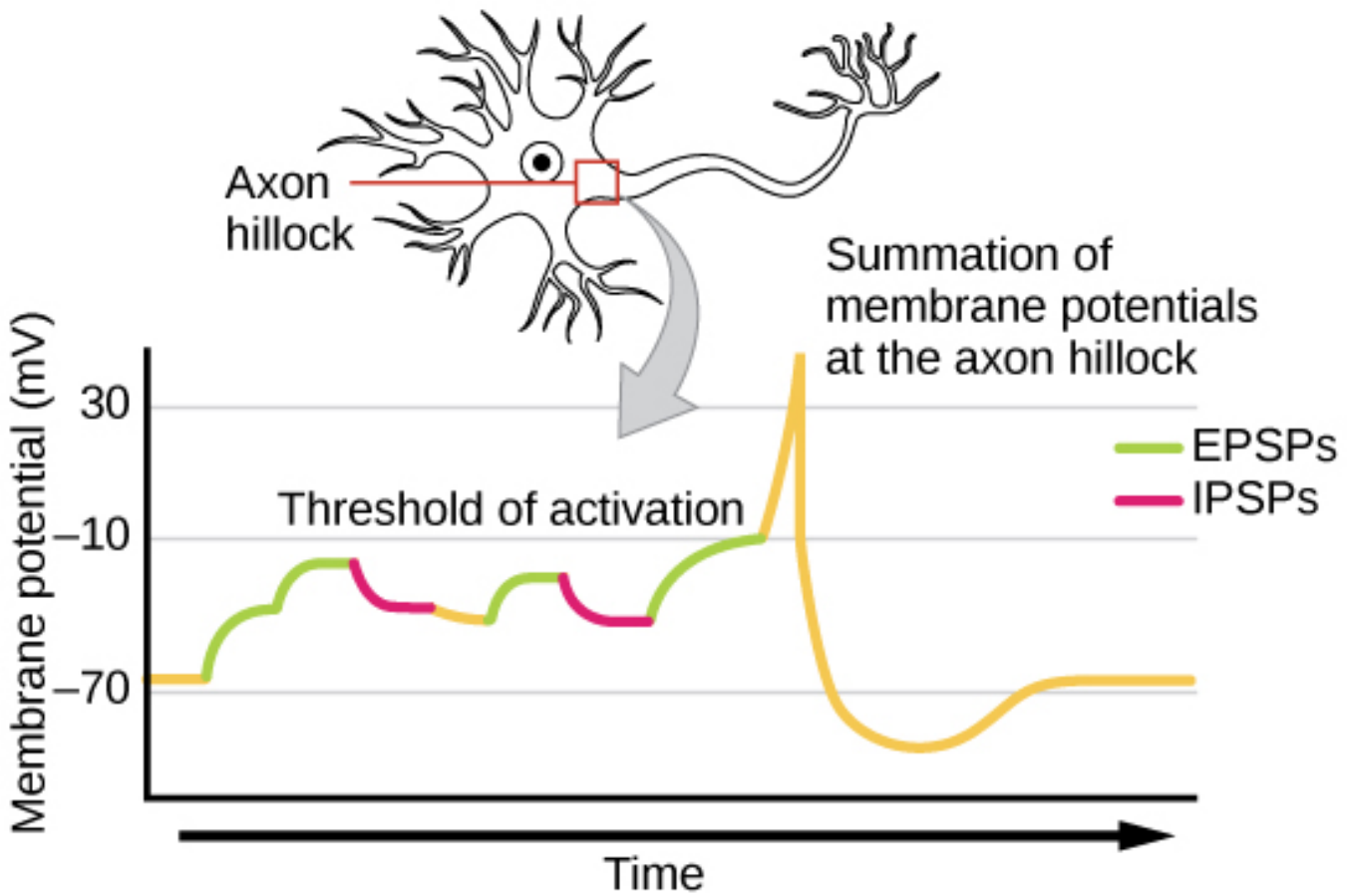
Similarities

$$y_1 = \sigma(w_1 x_1 + w_2 x_2 + w_3 x_3 + w_4)$$



- The excitation or firing of a biological neuron can be equated to a high positive value of units (x_1, x_2, x_3) in

Differences



- Biological neuron is all or None
- Biological neuron has a time component