Walk through + Neural Networks

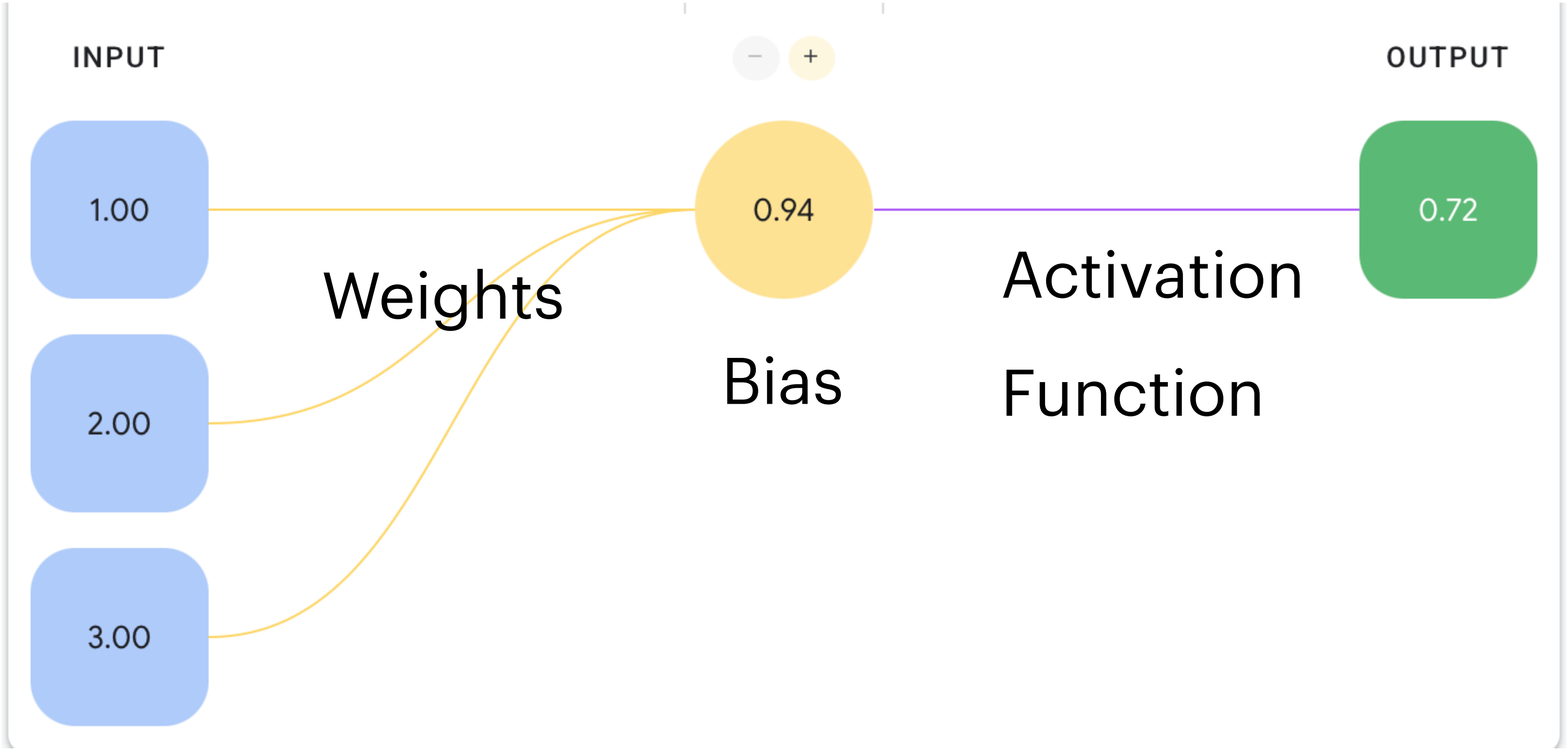
Divij 2025-02-26

The process

- Let there be some ***feature(s)*** x
- Let there be some ***outcome*** y
- Then, there exist some ***hypothesis*** h of the relationship between x and y
- This hypothesis relies on ***parameters*** K
- Using some ***loss*** L that we try to minimize, we estimate the best ***parameters*** for the given ***hypothesis***

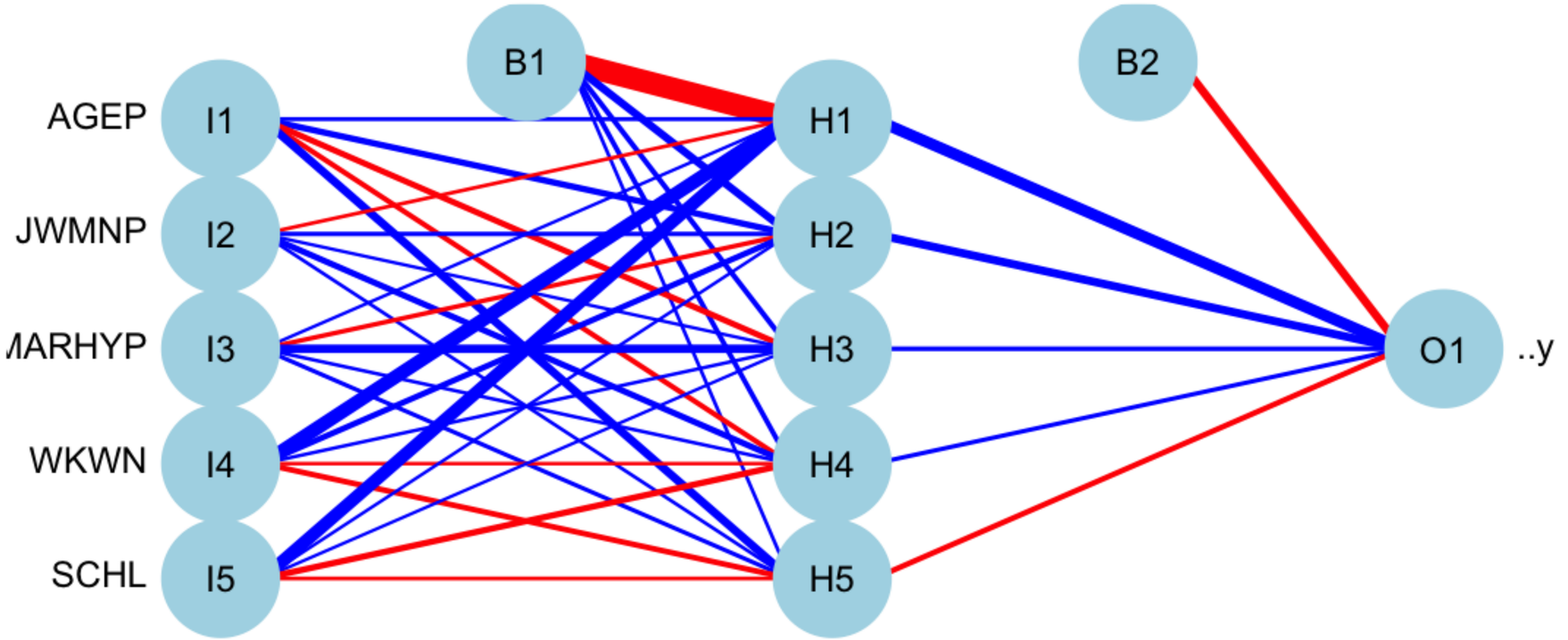
Why do this?

- Every model so far has the hypothesis decided
- What if we do not know the hypothesis???
- Neural networks can function as general learning machines, only limited by the complexity of the network



- https://developers.google.com/machine-learning/crash-course/neural-networks/nodes-hidden-layers

- **Weights**: These are the parameters that determine the strength of connections between neurons
- **Biases**: Biases are additional parameters that allow neurons to “shift” even when their inputs are zeros. They shift the activation function left or right.
- **Activation Functions**: These are mathematical operations applied to the weighted sum of inputs (plus bias) to determine a neuron's output. They introduce *non-linearity*, allowing neural networks to learn complex patterns.


B1
B2
AGEP
I1
H1
JWMNP
I2
H2
/ARHYP
I3
H3
O1
..y
WKWN
I4
H4
SCHL
I5
H5

How to train?

Back propagation!

- $NEW\ w_{11}^{0} = OLD\ w_{11}^{0} - \eta \frac{\delta L}{\delta w_{11}^{0}}$
  - η - Learning rate
    - How much of this to remember vs how much of the past to remember

- Repeat for all variables!

In practice

- Data **MUST** be scaled / normalized - neural networks are very sensitive to scale of data - we try to center around 0, and keep most values between -1 and 1
- Handle missing values or bad data before training the model

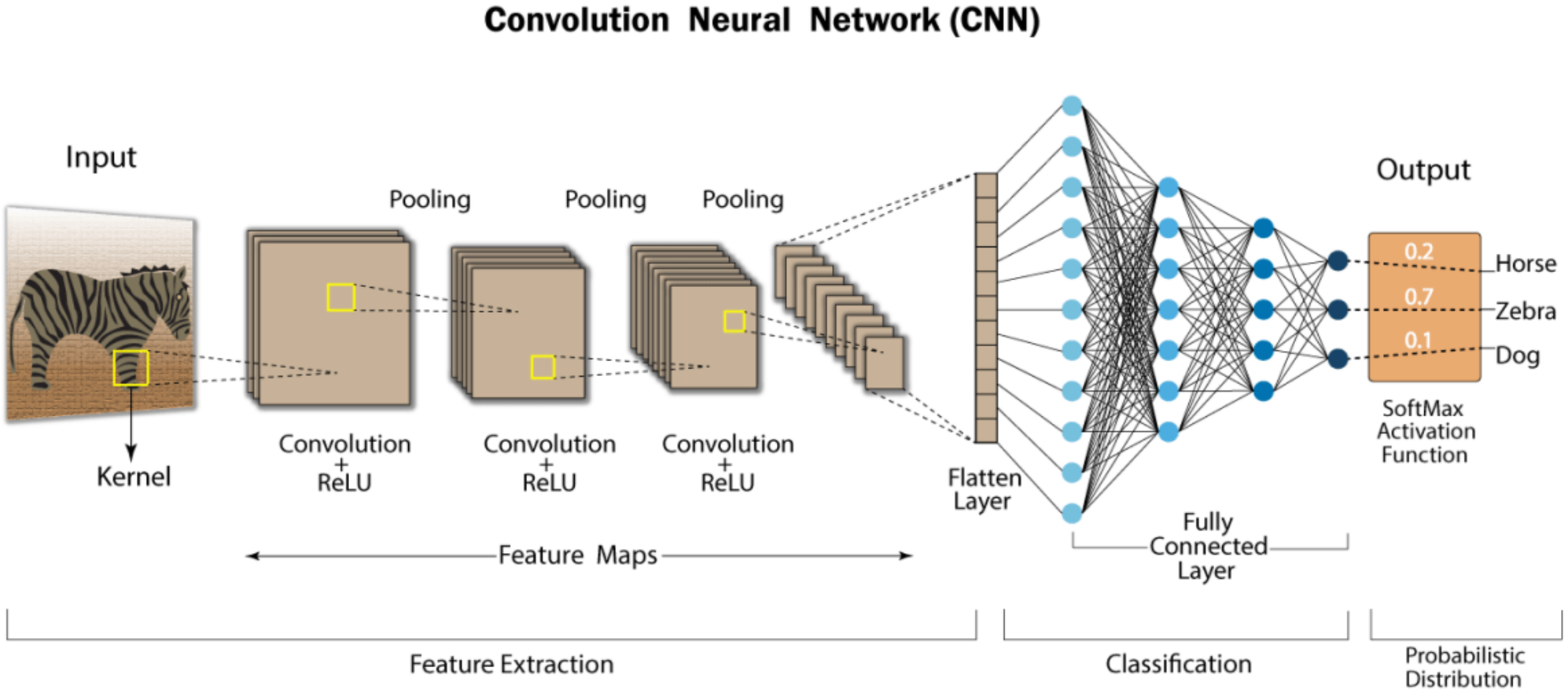
In practice

- Do as many calculations in parallel as possible!
- We move the loss only some some N number of inputs - Batch Size.
- We want to avoid the W/B getting too large or too small -> we ensure that the gradients are in a reasonable range
- We change the learning rate more dynamically, take larger steps earlier then smaller steps, then take some larger steps
- We also do not calculate the gradient every time, do it randomly - stochastic
  - This avoids getting stuck in “local minimas”, generally way faster, and computationally more efficient

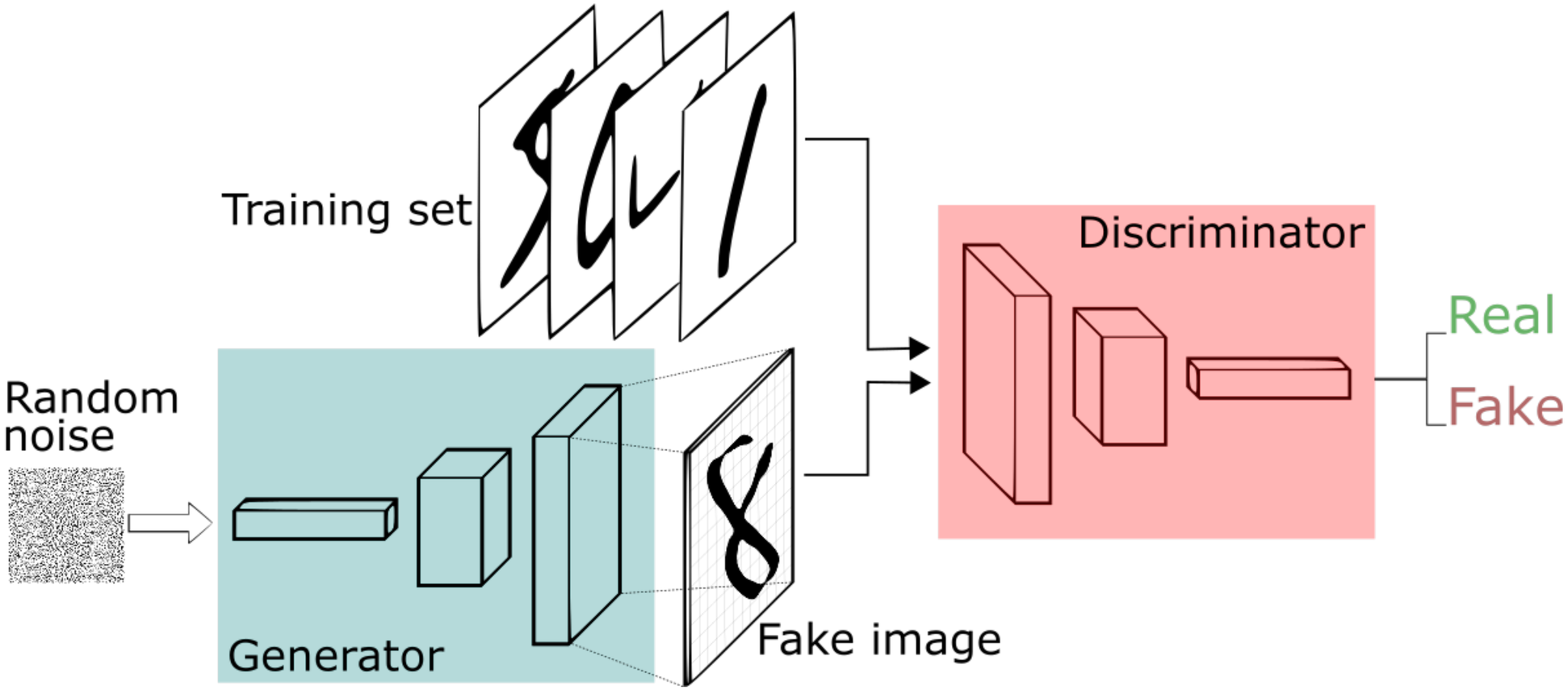
- We are now at the point where our understanding of future networks will not keep pace with what we can reasonably run on our systems
- Therefore, we can classify the ML we now know to be of two kinds
  - Ones we can model, train, and use all by ourselves
  - Ones where we rely on others to model, and train, and we end up being consumers of the models
- We need to learn rules of how to use these LLMs, what works well what does not etc etc.

Other popular architectures

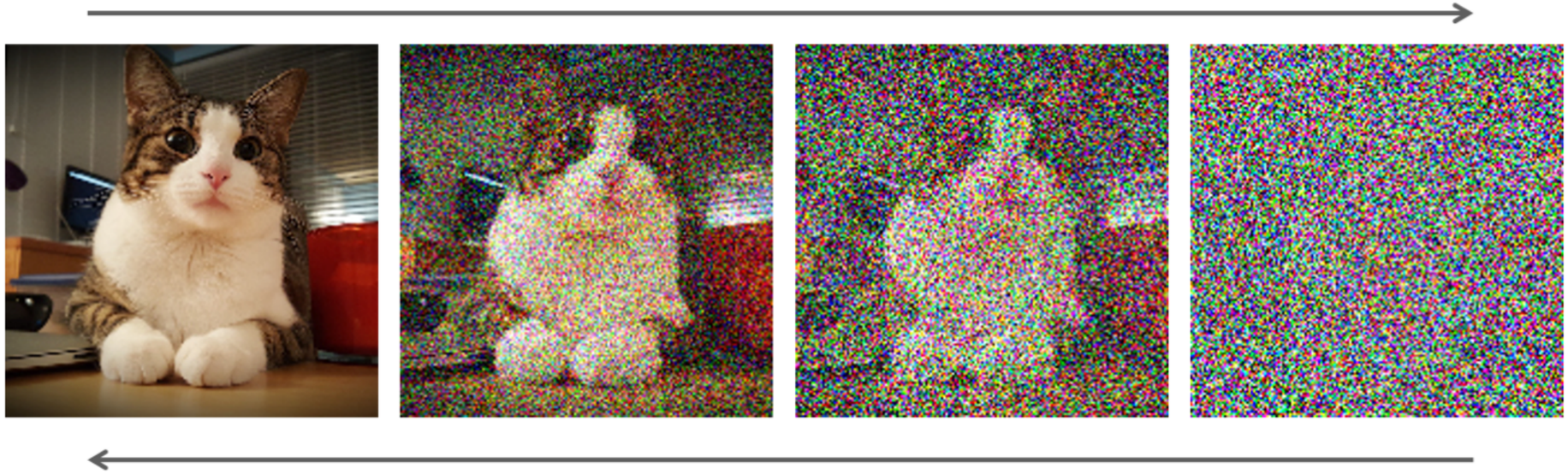
Convolutional Neural Networks



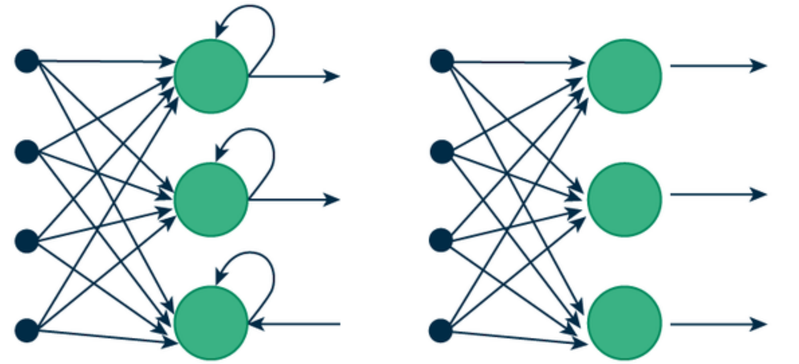
Generative Adversarial Networks



Diffusion Models



Recurrent Neural Networks



(a) Recurrent Neural Network (b) Feed-Forward Neural Network

A Simple RNN Language Model

output distribution

$$\hat{\boldsymbol{y}}^{(t)} = \text{softmax}\left(\boldsymbol{U}\boldsymbol{h}^{(t)} + \boldsymbol{b}_2\right) \in \mathbb{R}^{|V|}$$

Core idea: Apply the same weights *W* repeatedly

hidden states

$$\boldsymbol{h}^{(t)} = \sigma\left(\boldsymbol{W}_h\boldsymbol{h}^{(t-1)} + \boldsymbol{W}_e\boldsymbol{e}^{(t)} + \boldsymbol{b}_1\right)$$

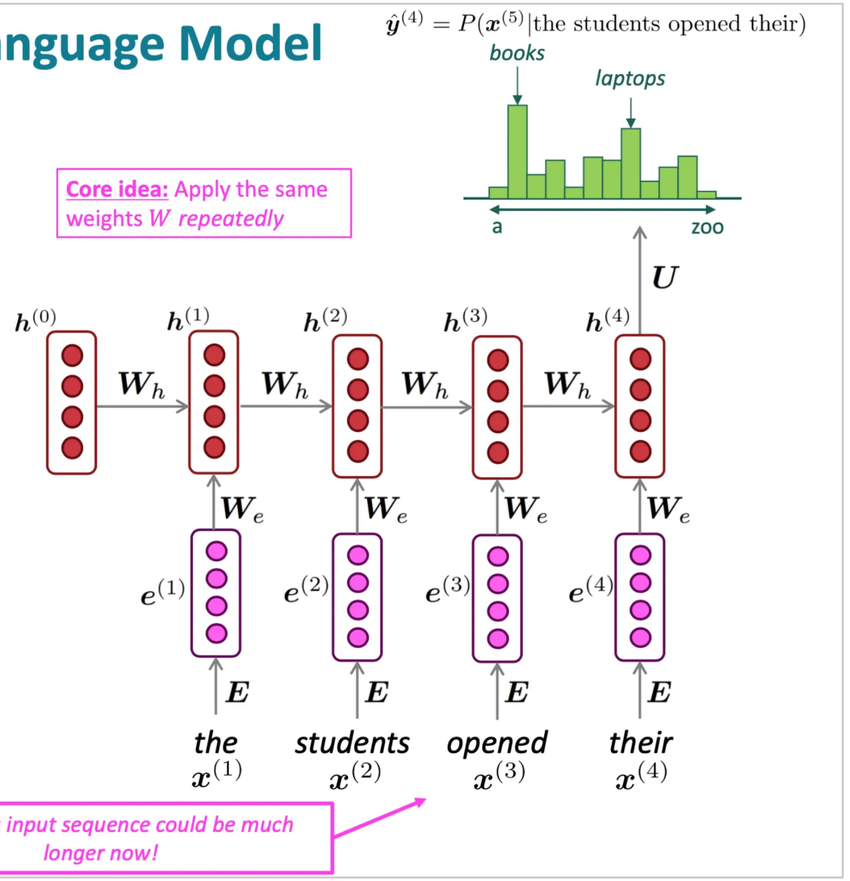
$\boldsymbol{h}^{(0)}$ is the initial hidden state

word embeddings

$$\boldsymbol{e}^{(t)} = \boldsymbol{E}\boldsymbol{x}^{(t)}$$

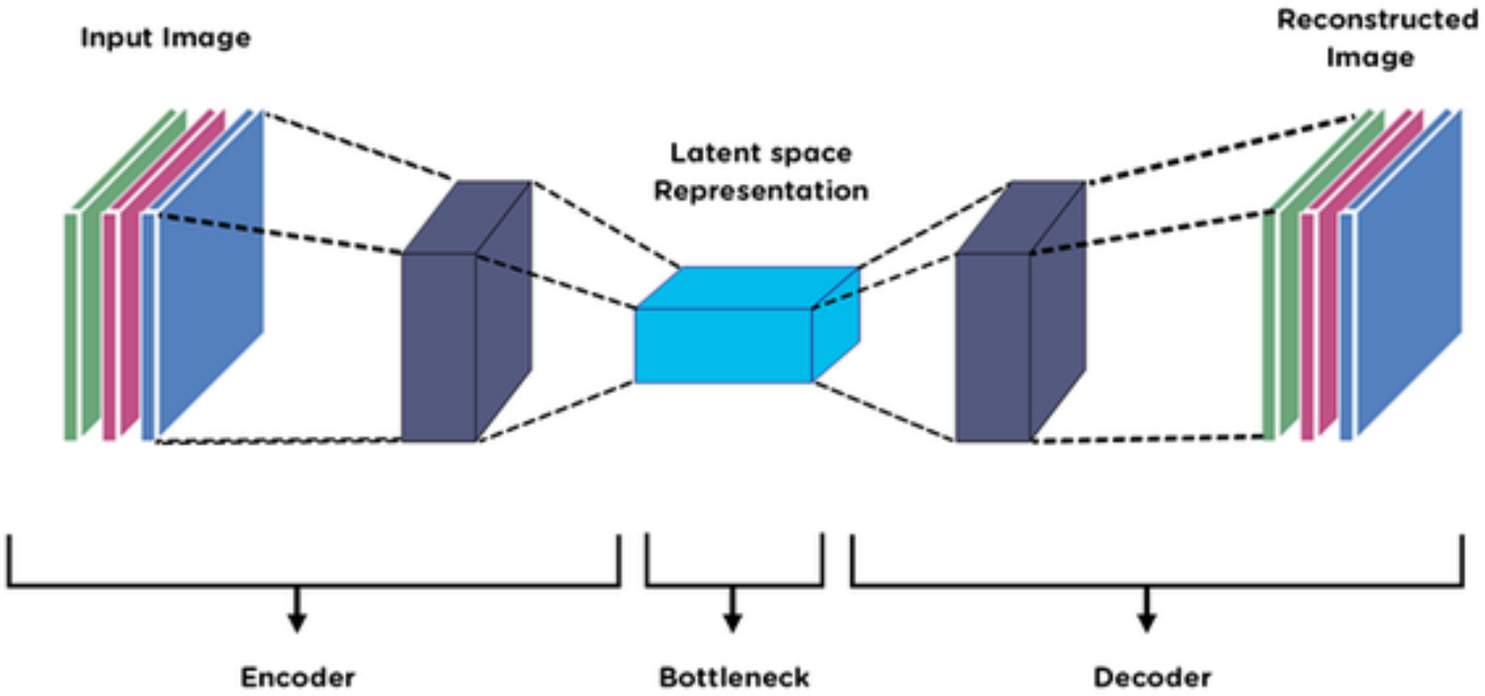
words / one-hot vectors

$$\boldsymbol{x}^{(t)} \in \mathbb{R}^{|V|}$$



Note: this input sequence could be much longer now!

Auto encoders



Transformers

ENCODER ONLY

aka

auto-encoding models

TASKS

- Sentence classification
- Named entity recognition
- Extractive question-answering
- Masked language modeling

EXAMPLES

BERT, RoBERTa, distilBERT

DECODER ONLY

aka

auto-regressive models

TASKS

- Text generation
- Causal language modeling

EXAMPLES

GPT-2, GPT Neo, GPT-3

ENCODER-DECODER

aka

sequence-to-sequence models

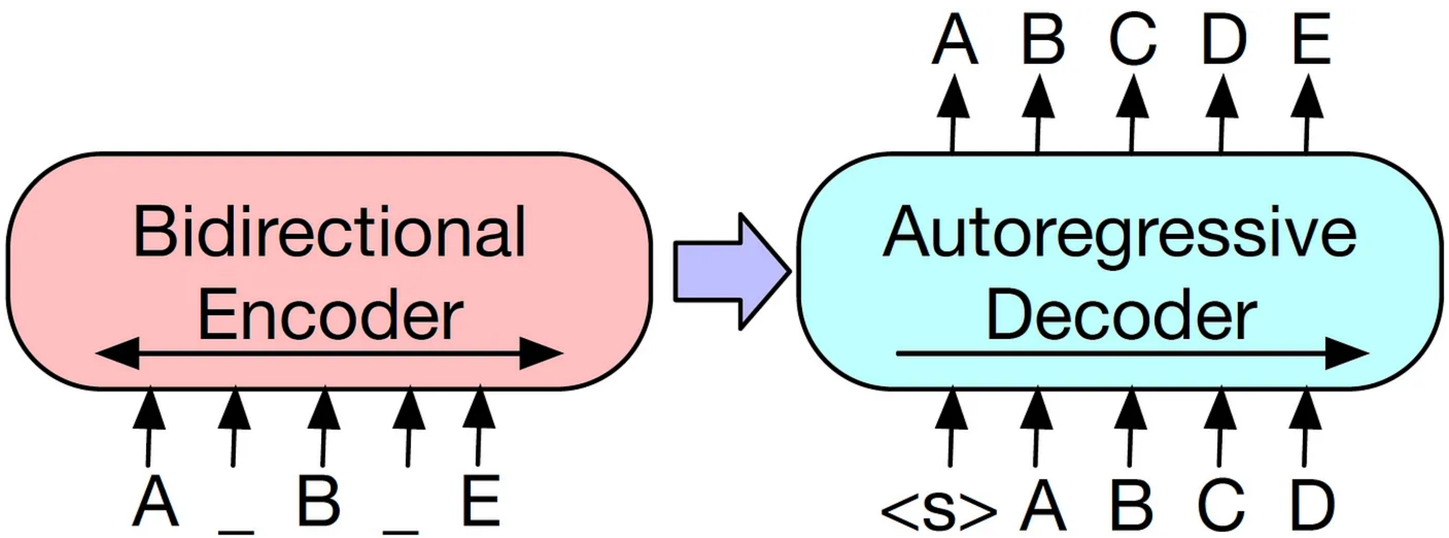
TASKS

- Translation
- Summarization
- Generative question-answering

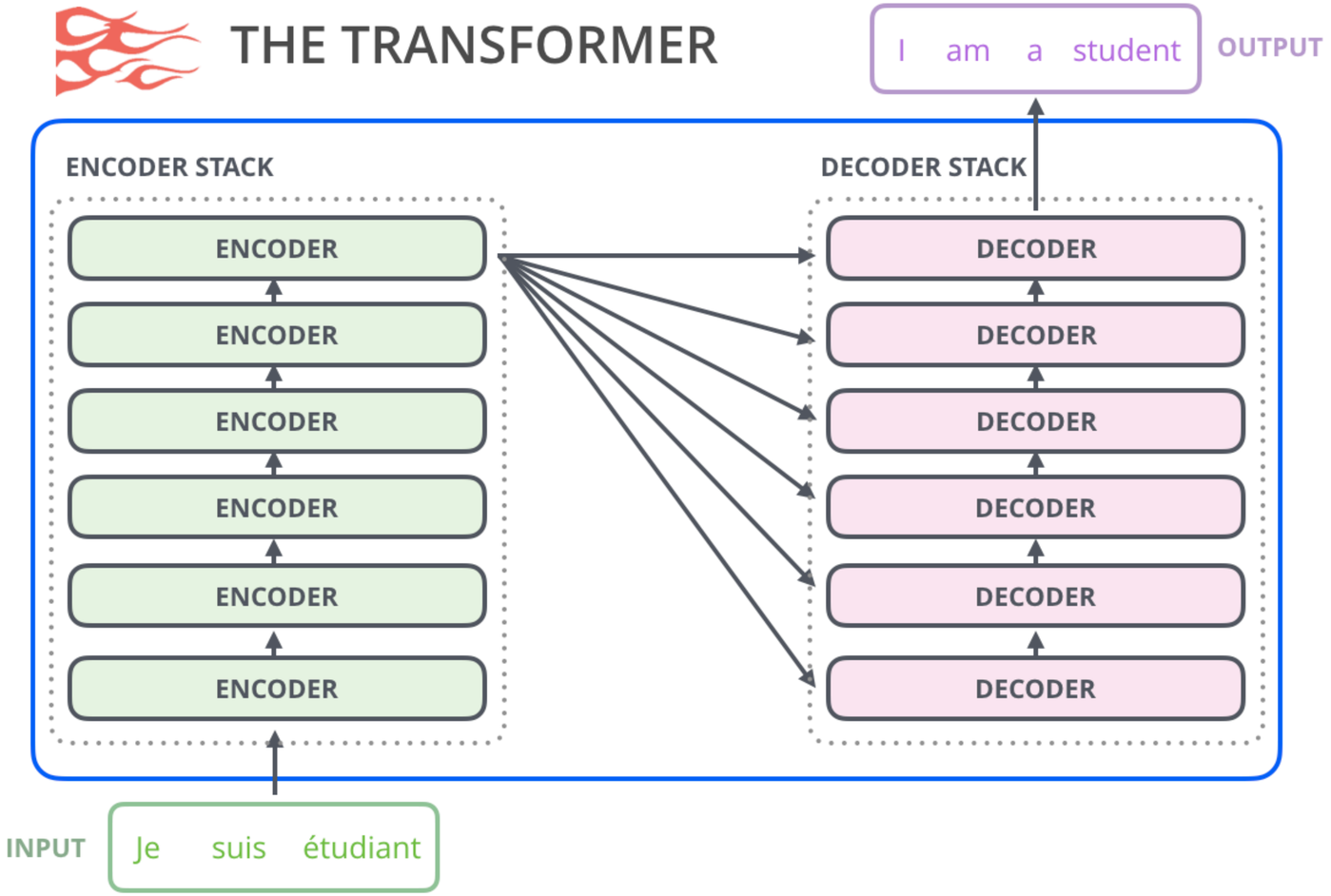
EXAMPLES

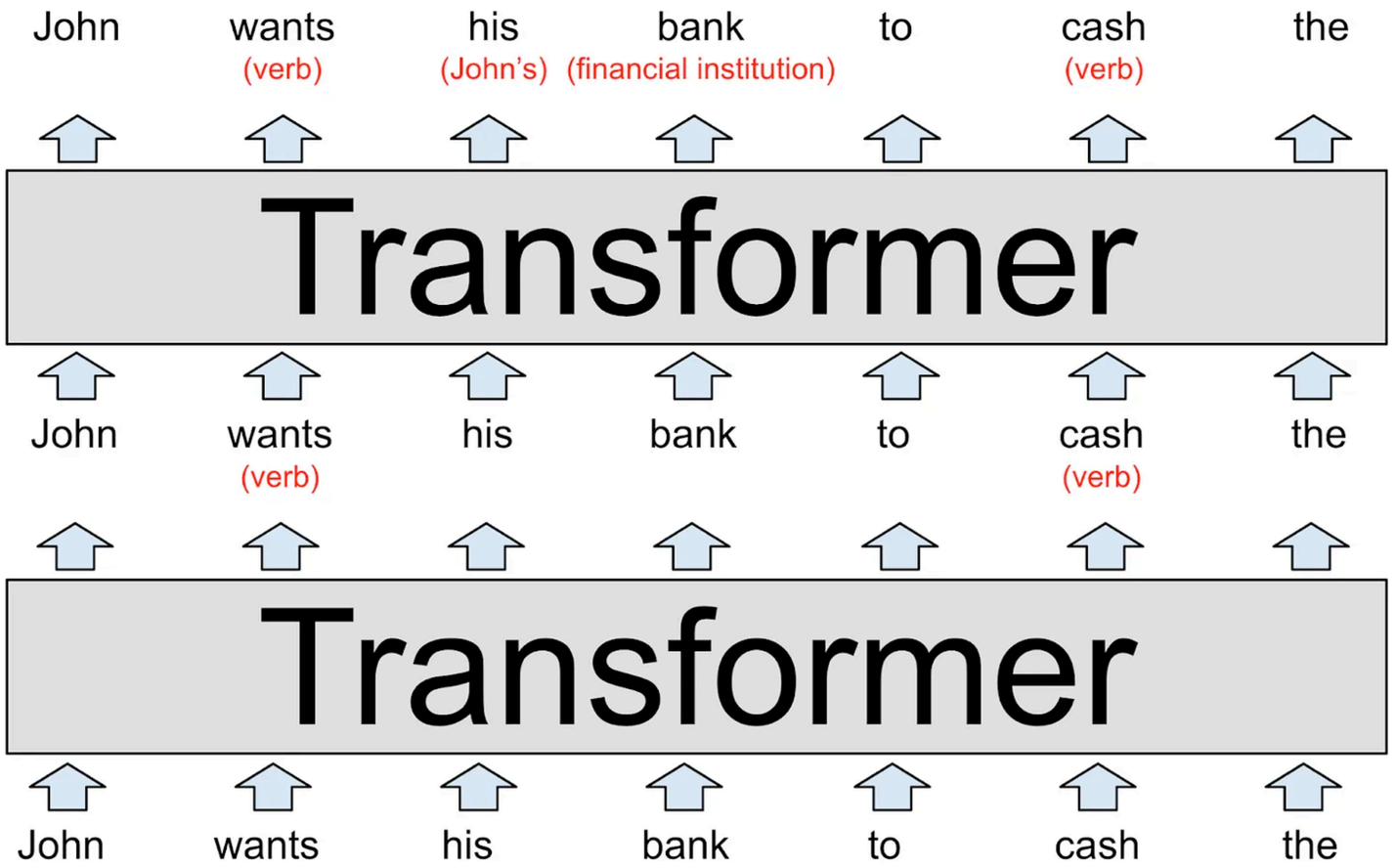
BART, T5, Marian

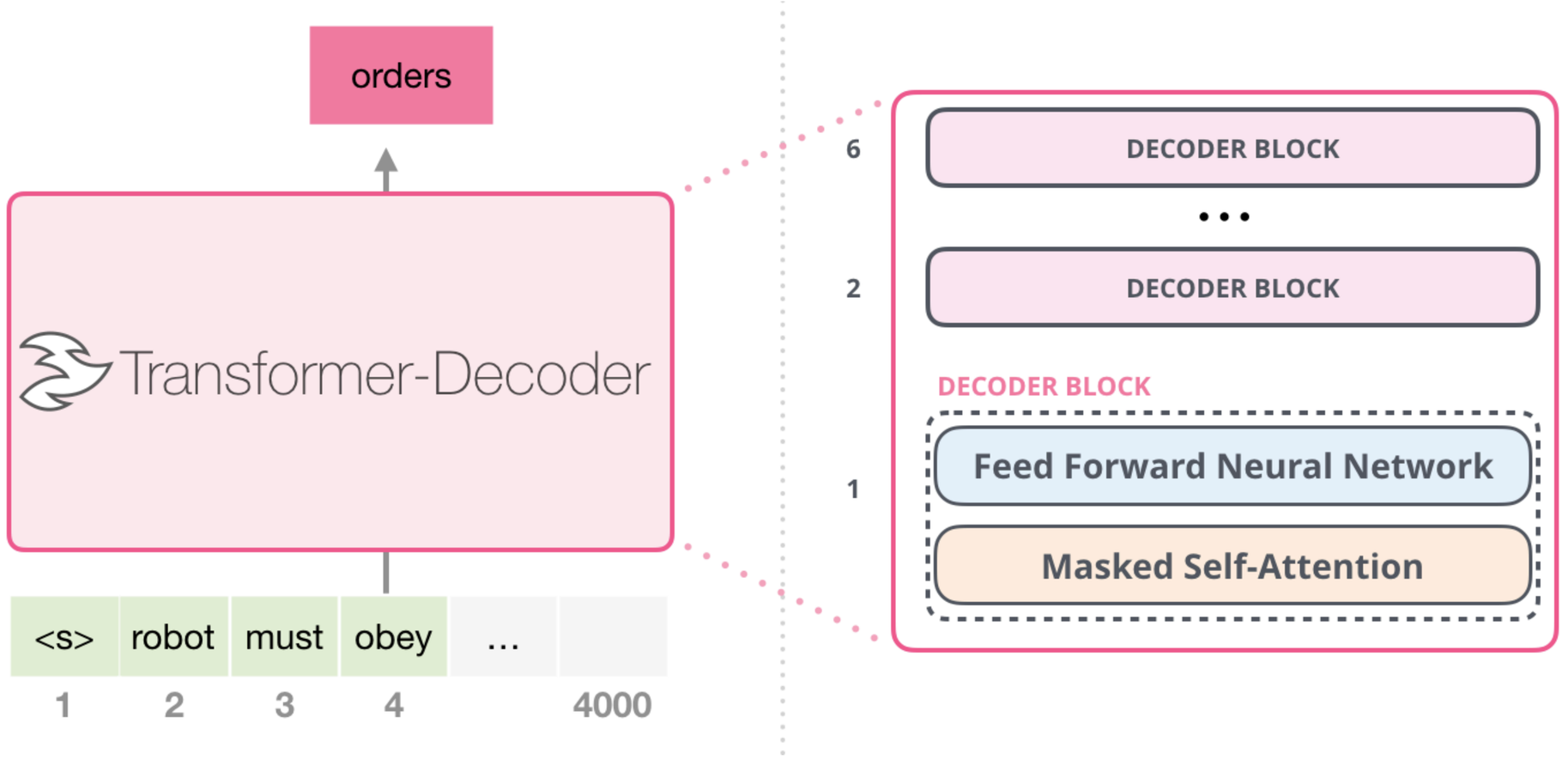
Transformers

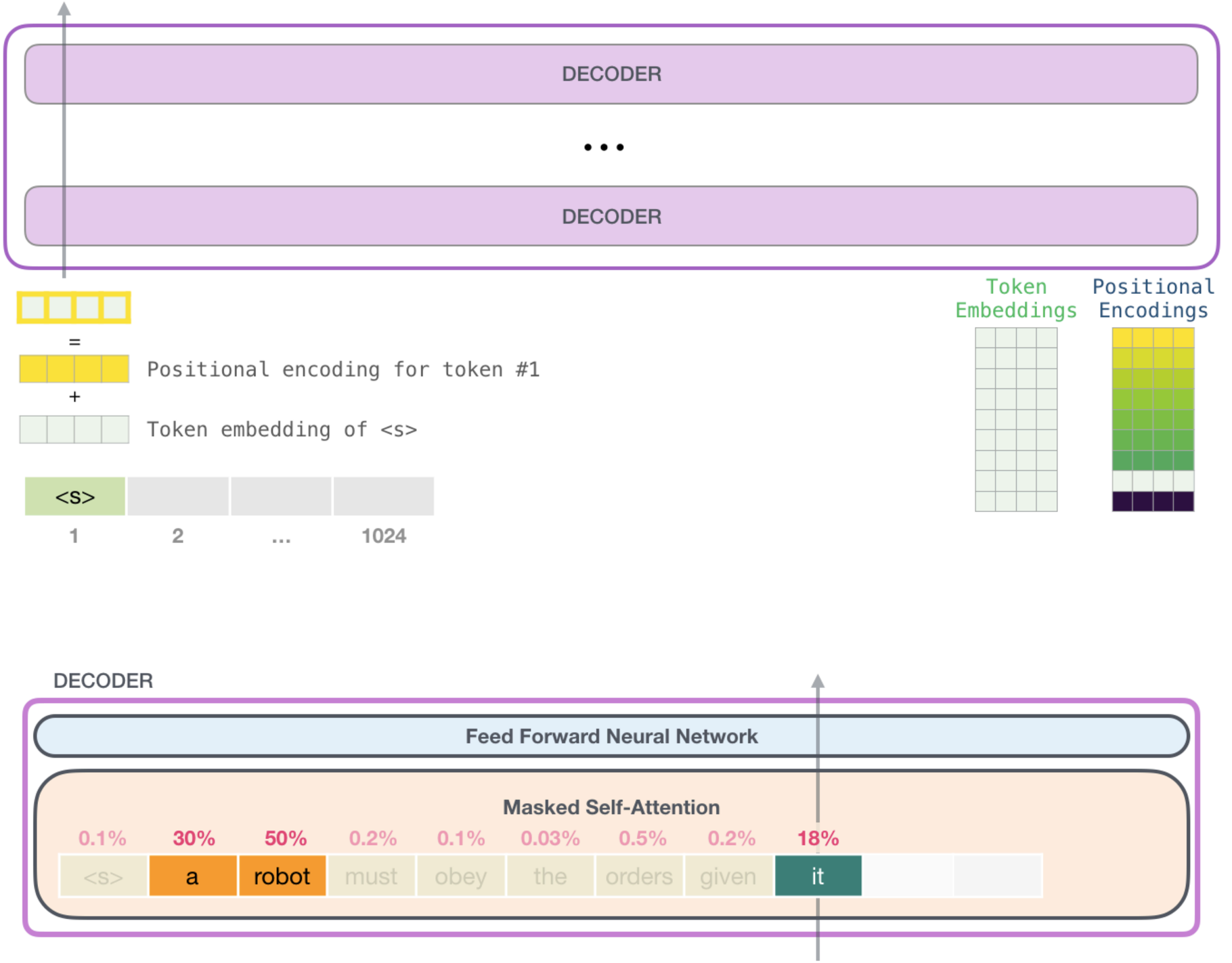


Transformers

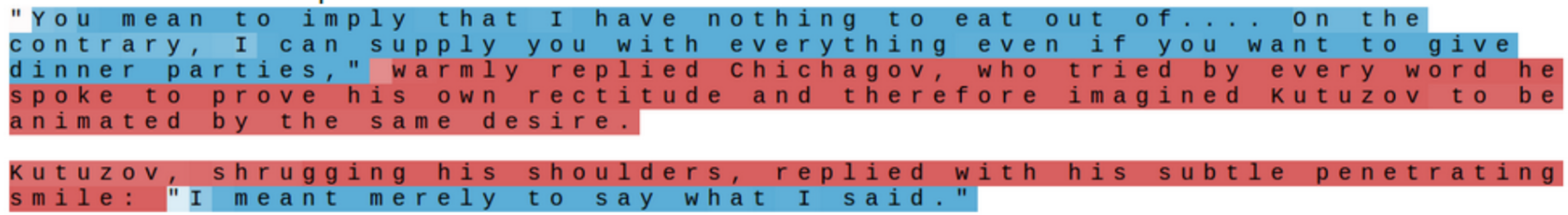



John
wants
(verb)
his
(John's)
bank
(financial institution)
to
cash
(verb)
the
Transformer
John
wants
(verb)
his
bank
to
cash
(verb)
the
Transformer
John
wants
his
bank
to
cash
the

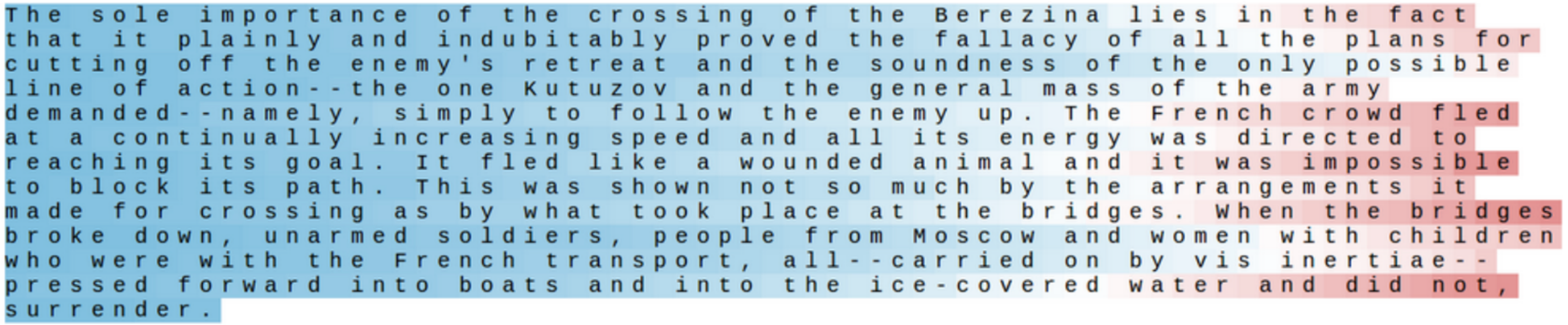

orders
Transformer-Decoder
<s>
robot
must
obey
...
1
2
3
4
4000
6
2
1
DECODER BLOCK
DECODER BLOCK
DECODER BLOCK
Feed Forward Neural Network
Masked Self-Attention


DECODER
DECODER
Token Embeddings
Positional Encodings
=
Positional encoding for token #1
+
Token embedding of <s>
<s>
1
2
...
1024
DECODER
Feed Forward Neural Network
Masked Self-Attention
0.1%
30%
50%
0.2%
0.1%
0.03%
0.5%
0.2%
18%
<s>
a
robot
must
obey
the
orders
given
it

Cell that turns on inside quotes:



Cell sensitive to position in line:



Current state of LLMs

- LLMs seem to be good at quite general purpose tasks.
- Their general structure is outputting the next “token” or “word”
- They use a few tricks and slightly different structures than we are used to
- The question is - how do we leverage what we know about the learning process to have better outputs?
  - Garbage in, garbage out
  - Gold in, gold out

Ways to use LLMs in practice

- Run open models on your own laptop - needs investment, slow, but private
- Run open models on servers - continuing cost, somewhat private
- Use models hosted by others - cheaper, fast, but generally speaking not very private
  - Varying degrees

Demo time

- API
- BERT
- Census App

What about reasoning?

- Being descriptive helps in the input process, it should therefore also help in the output process

“Normal”

Training data example

- <user> What is the number of “r” in the word “strawberry” </user>
  <agent>There are 3 “r” in strawberry </agent>
- <user> What is the number of “r” in the word “banana” </user>
  <agent>There are 3 “r” in banana </agent>

w/ Reasoning

Training data example

- <user> What is the number of “r” in the word “strawberry” </user>
  <thinking> “strawberry” has the following letters “s” “t” “r” “a” “w” “b” “e” “y”
  Their frequencies are: “s” - 1 ; “t” - 1 ; “r” - 3 ; “a” - 1 ; “b” - 1 ; “e” - 1 ; “y” - 1
  </thinking>
  <agent>There are 3 “r” in strawberry </agent>
- <user> What is the number of “r” in the word “banana” </user>
  <thinking> “banana” has the following letters “b” “a” “n”
  Their frequencies are: “b” - 1 ; “a” - 3 ; “n” - 2
  </thinking>
  <agent>There are 0 “r” in banana </agent>

Which do you think will perform better

- Generate some code for me - I want to use R. The context of the code is a single file qmd that runs through a simple ML model, that includes loading in a dataset, cleaning the data, visualising it, and then coming up with a couple of different simple models like random forest, or logistic regression, or linear regression etc, to try and model an output. It also should talk through the metrics for the models, and figure out which is the best one. Ideally including some sort of hyper parameter tuning as well. It should use GGplot, tidymodels, tidyverse, and maybe tidycensus/tigris/sf if needed.

- Create an educational R Quarto (qmd) document for a "ML and AI for Public Policy" course aimed at social science students. This document should serve as a teaching tool that demonstrates how machine learning can be applied to policy-relevant problems.
- Use the "ACS Public Use Microdata Sample" data from the tidycensus package to predict household income levels based on demographic and socioeconomic factors. This example will show students how ML can be used to identify patterns and potential disparities in economic outcomes.
- The document should:
- 1. Introduction:
- - Frame the analysis in terms of policy relevance (e.g., understanding income inequality, targeting assistance programs)
- - Explain ethical considerations when using ML for policy decisions
- - Discuss how predictive models can inform evidence-based policymaking
- - List and explain the packages we'll use (tidyverse, tidymodels, ggplot2, tidycensus)
- 2. Data acquisition and understanding:
- - Show how to access and load census data using tidycensus
- - Discuss the importance of understanding government data sources
- - Explain sampling methodology and limitations of the data
- - Connect the dataset to real-world policy questions
- 3. Data cleaning and preparation:
- - Handle missing values with discussion of how missing data may introduce bias
- - Transform variables as needed (income brackets, categorical variables)
- - Create derived features relevant to policy analysis
- - Discuss how data preparation choices might impact different demographic groups
- 4. Exploratory analysis through a policy lens:
- - Create visualizations showing income distributions across geographic regions or demographic groups
- - Explore relationships between education, employment, demographics, and income
- - Discuss how visualizations can effectively communicate findings to policymakers
- - Include maps using sf for geographic analysis of disparities
- 5. Feature selection and engineering:
- - Discuss which variables are appropriate to include in a policy context
- - Address potential sources of bias in feature selection
- - Create policy-relevant interaction terms
- - Explain the balance between model complexity and interpretability for policymakers

- 6. Model building:
  - Split data with discussion of representative sampling
  - Implement multiple models with increasing complexity:
    - Linear Regression (baseline for interpretability)
    - Random Forest
    - Boosted Trees
  - Discuss tradeoffs between model complexity and interpretability in policy contexts
- 7. Hyperparameter tuning:
  - Demonstrate tuning for Random Forest and Boosted Trees
  - Explain computational resource considerations for government applications
  - Visualize how parameter choices affect model performance
- 8. Model evaluation with policy implications:
  - Evaluate models using metrics relevant to policy decisions
  - Assess fairness across demographic groups
  - Analyze which features drive predictions (feature importance)
  - Discuss how to communicate model uncertainty to policymakers
- 9. Final model selection and policy application:
  - Select best model considering both performance and interpretability
  - Demonstrate how to use the model for policy scenario analysis
  - Discuss limitations and potential unintended consequences
  - Explore how the model could inform specific policy interventions
- 10. Ethical considerations and governance:
  - Address potential biases in the model
  - Discuss responsible AI principles for public sector use
  - Consider transparency requirements for algorithmic decision-making
  - Outline governance frameworks for ML in public policy
- Throughout the document, include "Policy Insight" sections that connect technical concepts to public policy applications and ethical considerations.