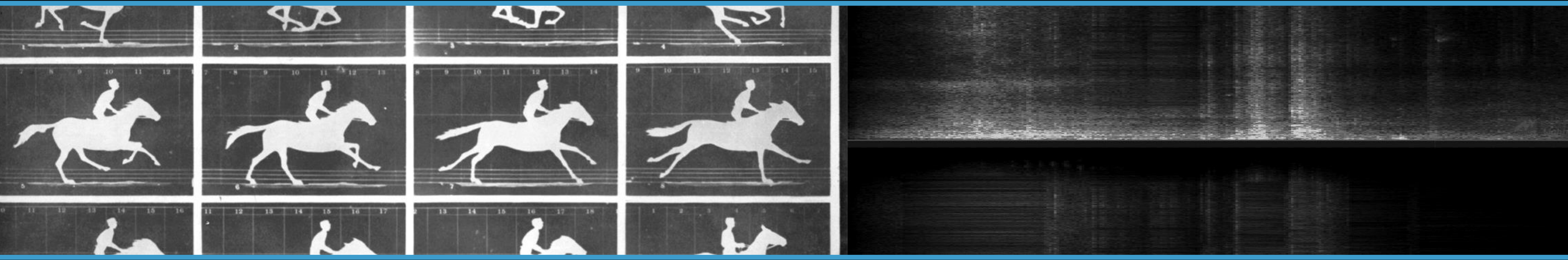
AI for the Media

Week 5, Classifying Sequences



~ Vít Růžička

Overview

Classifying Sequences (*pre-recorded lecture*):

- Sequential data properties and representations
- Sequential model schematics and training
- Classification, regression and generation

Practical session (*during the live session*):

- Code: sentiment analysis for tweets

Sequential data

- What properties does sequential data have?

Sequential data

- ~~What properties does sequential data have~~?
- How does non-sequential data look and how do we process it?

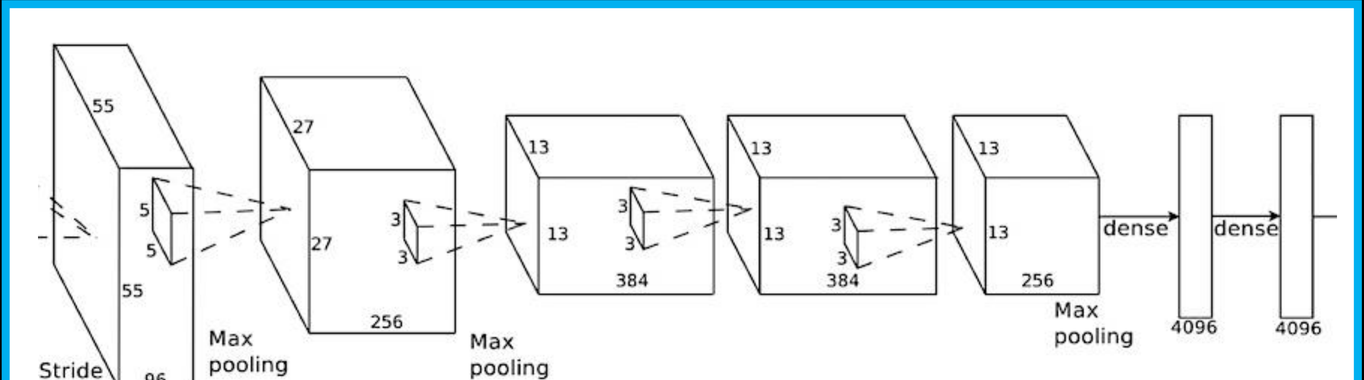
Sequential data

- ~~What properties does sequential data have~~?
- How does non-sequential data look and how do we process it?

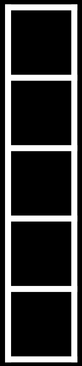
^ input = image
(*height*width*rgb* grid of pixels)

Sequential data

- ~~What properties does sequential data have~~?
- How does non-sequential data look and how do we process it?

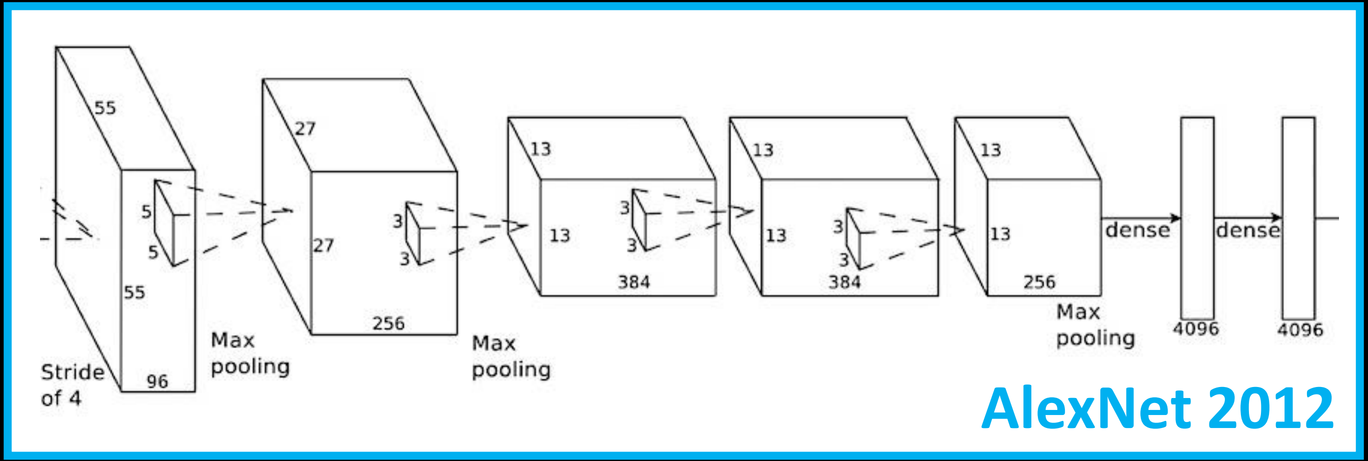


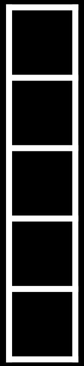
AlexNet 2012



Sequential data

- ~~What properties does sequential data have~~?
- How does non-sequential data look and how do we process it?





^ For each one input (*image*)

^ We have one output label

Sequential data

- What properties does sequential data have?

A	test	sentence	…

→ model → *Output feature (or classification)*

Sequential data

- What properties does sequential data have?

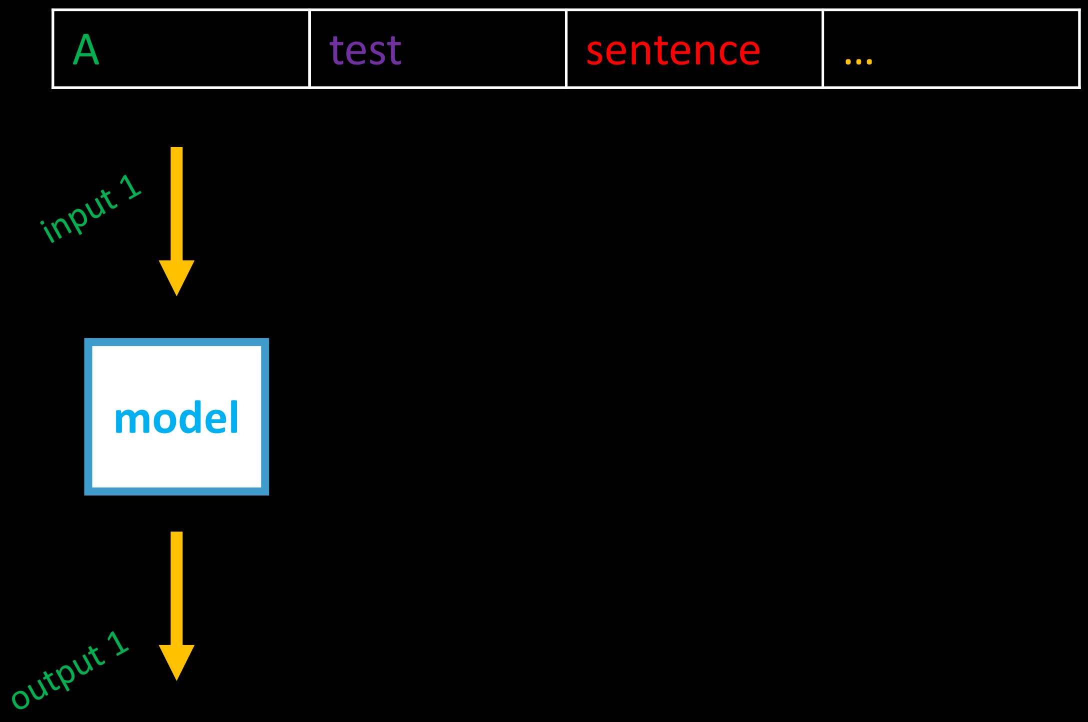
A	test	sentence	…

→ **model** → *Output feature (or classification)*

- **Order matters**

Sequential data

- What properties does sequential data have?

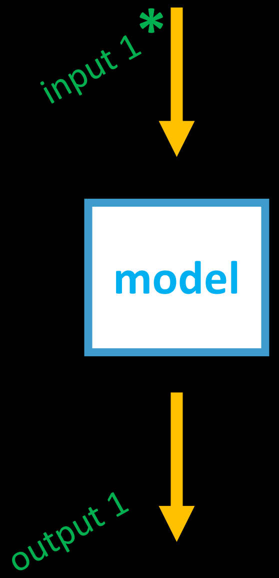


- **Order matters**
- We *want to* input the data in this order

Sequential data

- What properties does sequential data have?

A	test	sentence	…

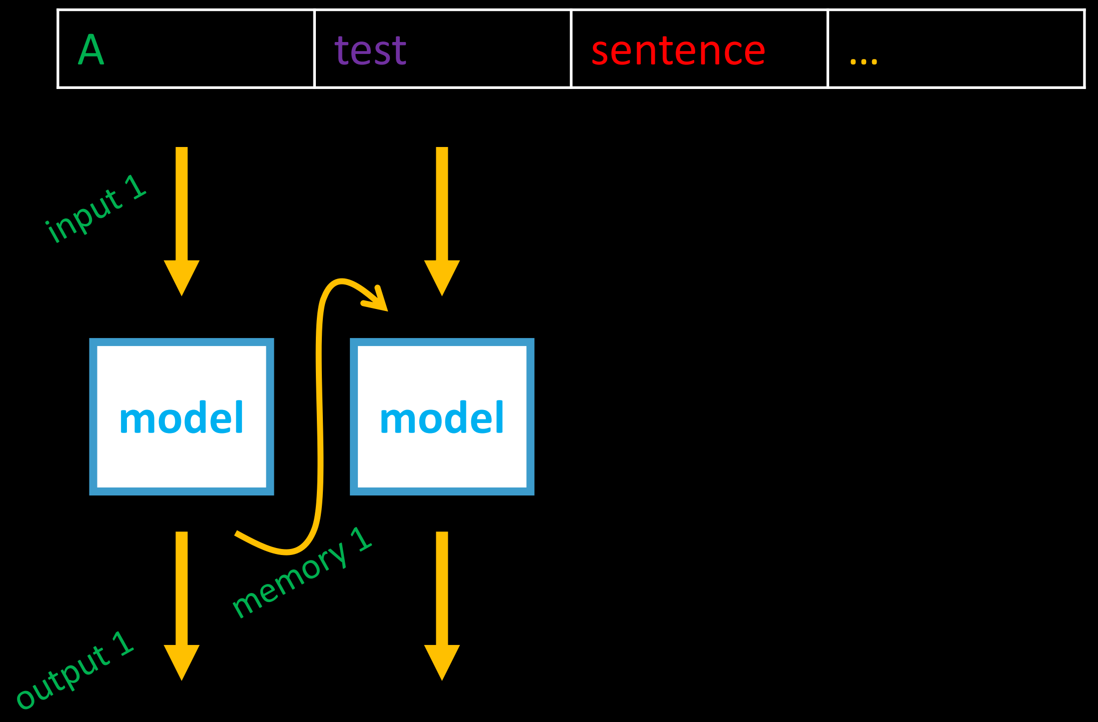


- **Order matters**
- We *want to* input the data in this order

*) You can probably already see that this would be some sort of representation of the word "*A*", maybe some vector we got from word2vec …

Sequential data

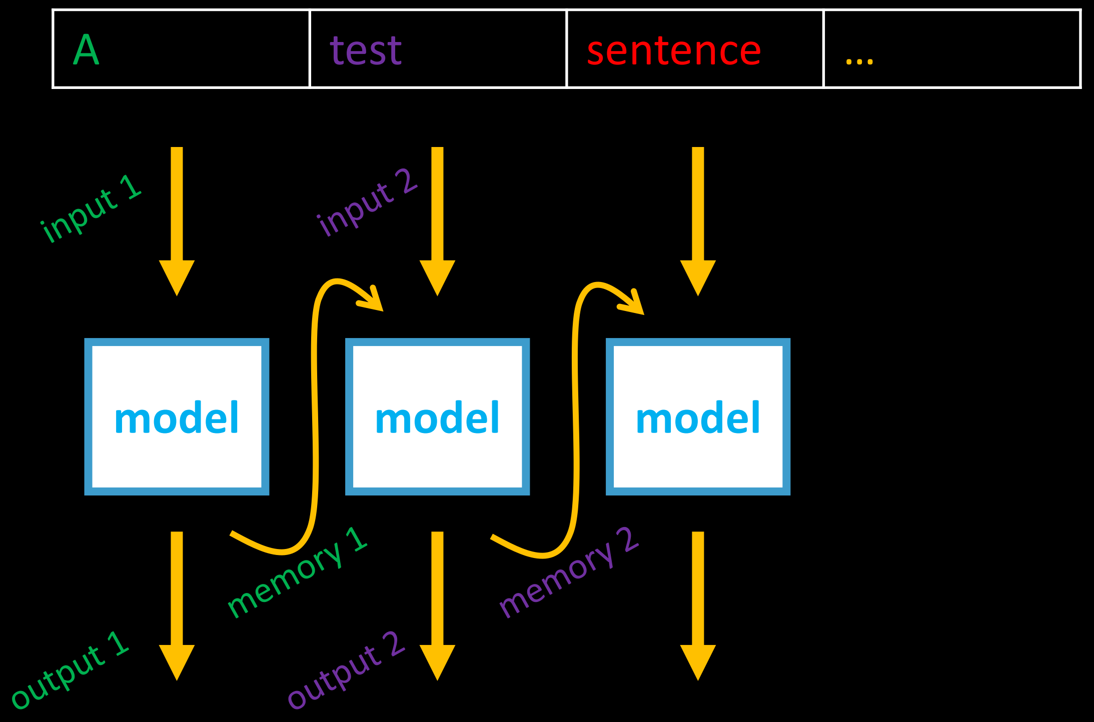
- What properties does sequential data have?



- **Order matters**
- We *want to* input the data in this order

Sequential data

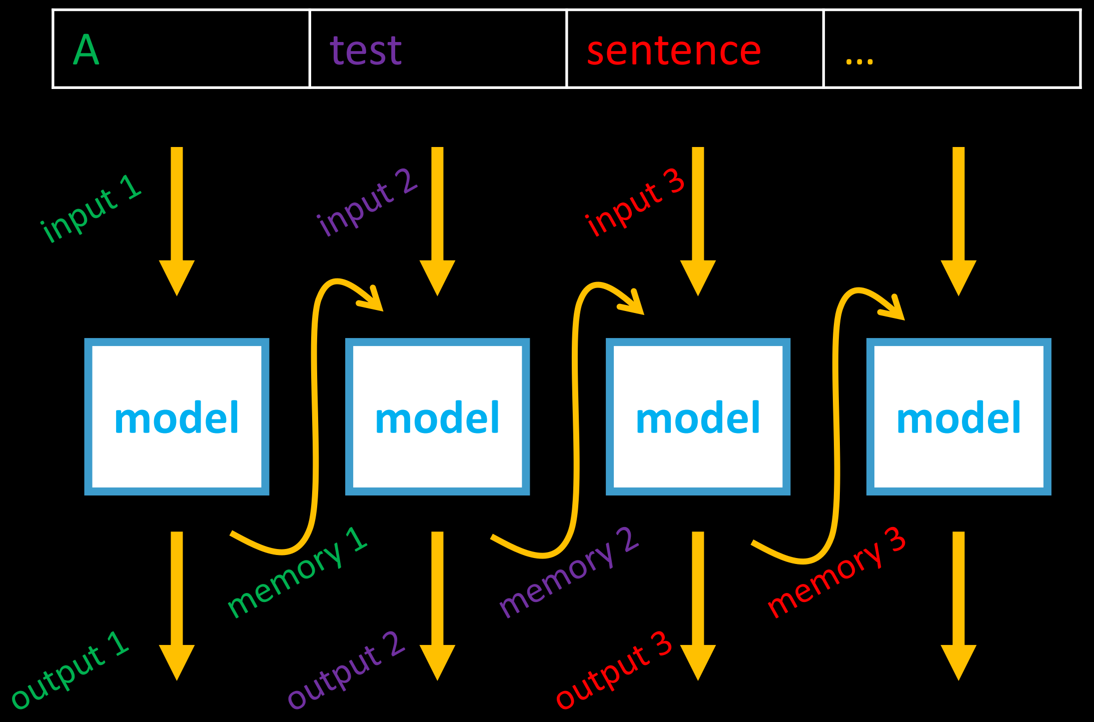
- What properties does sequential data have?



- **Order matters**
- We *want to* input the data in this order

Sequential data

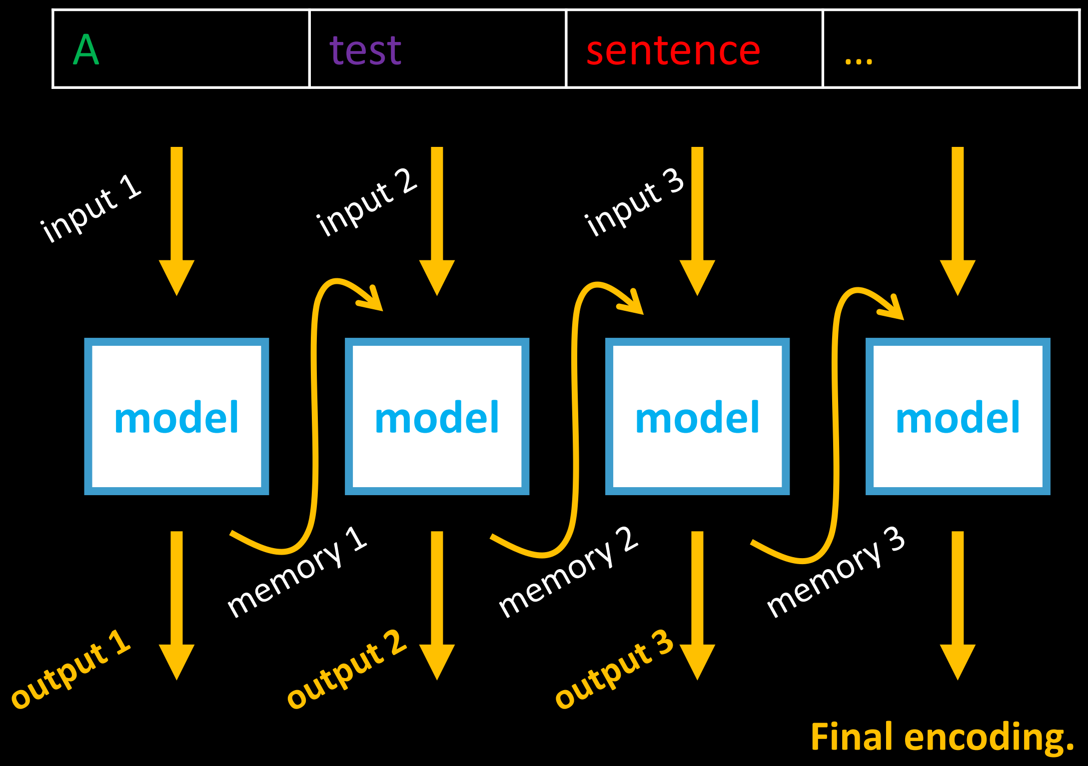
- What properties does sequential data have?



- **Order matters**
- We *want to* input the data in this order

Sequential data

- What properties does sequential data have?

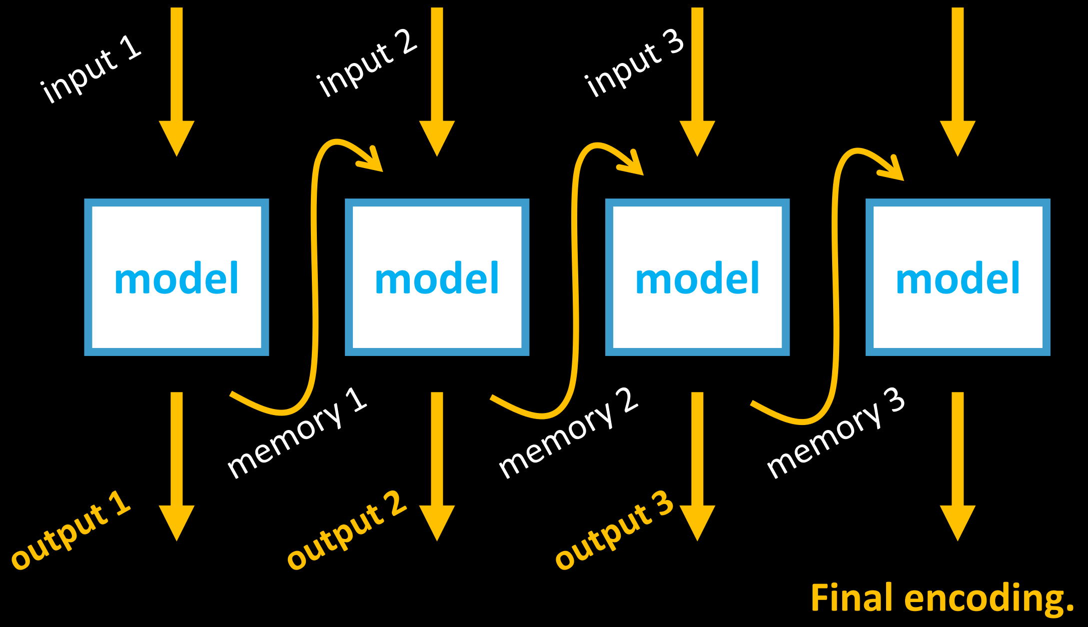


- **Order matters**
- We *want to* input the data in this order
- We get **intermediate states** after each input we feed in

Sequential data

- What properties does sequential data have?

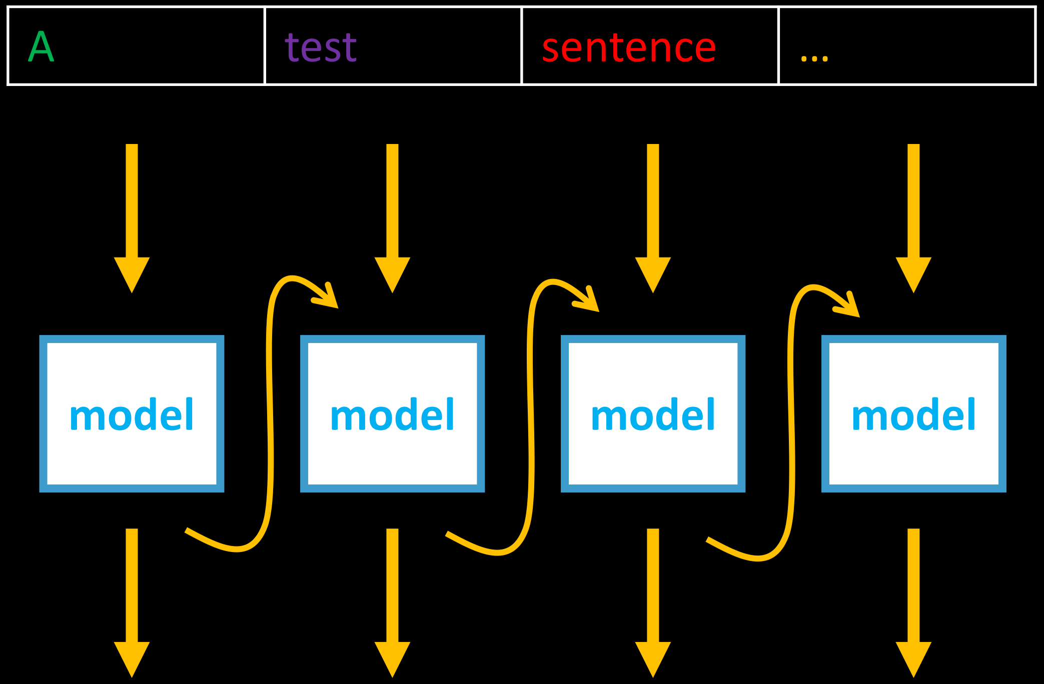
Short	sentence	.	
Longer	sentence	continuing	…



- **Order matters**
- We *want to* input the data in this order
- We get **intermediate states** after each input we feed in
- *Note: Special to sequential models, they allow for inputs of differing lengths.*

Sequential data

- What properties does sequential data have?

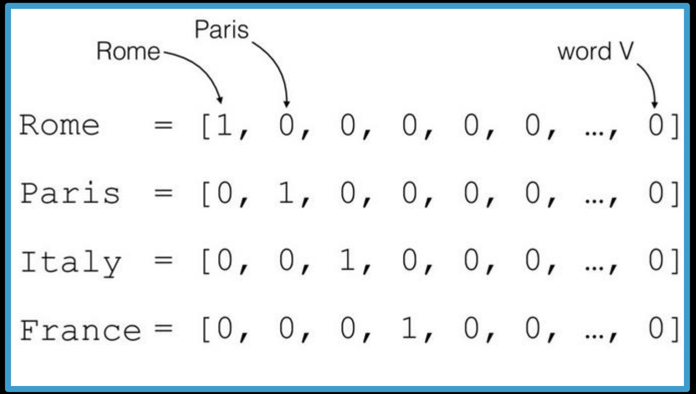


We want to build a **model** that considers the sequentiality of the data

Data representation

Text

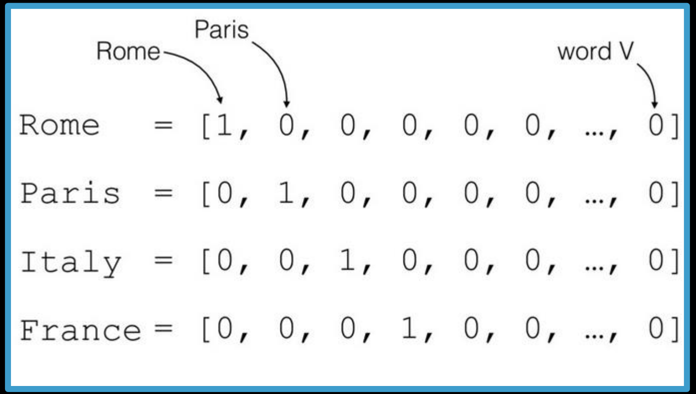
- One-hot vectors



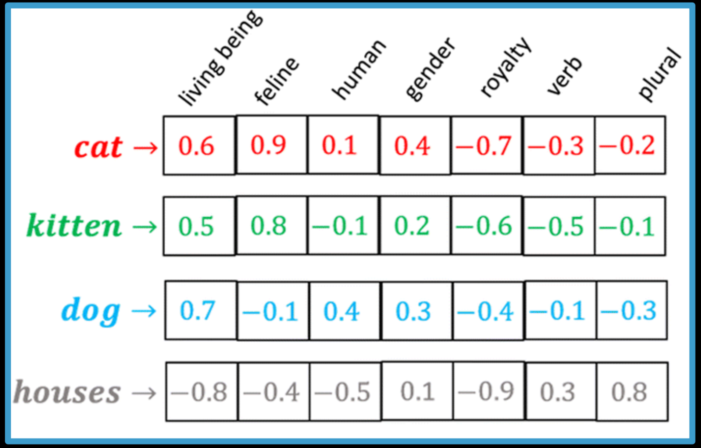
Data representation

Text

- One-hot vectors



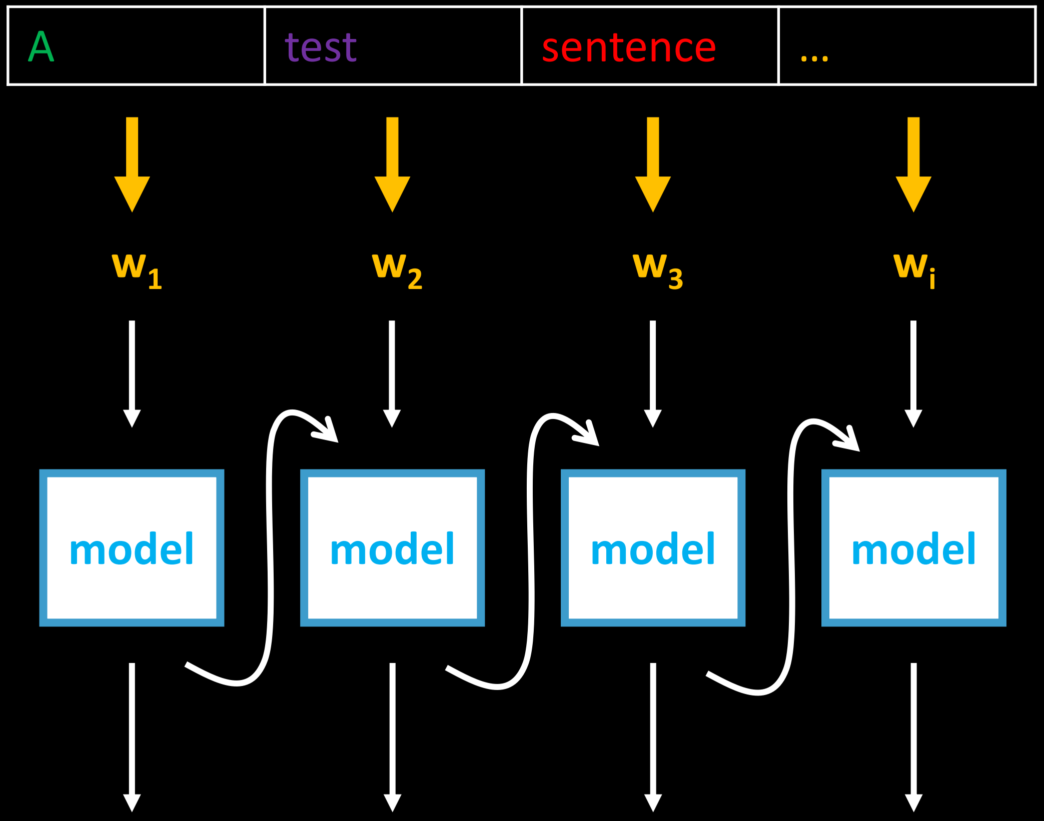
- Embeddings *"someone else"* gave us
  - Word2vec, GloVe embedding, etc…



- Our own embeddings from a model we train

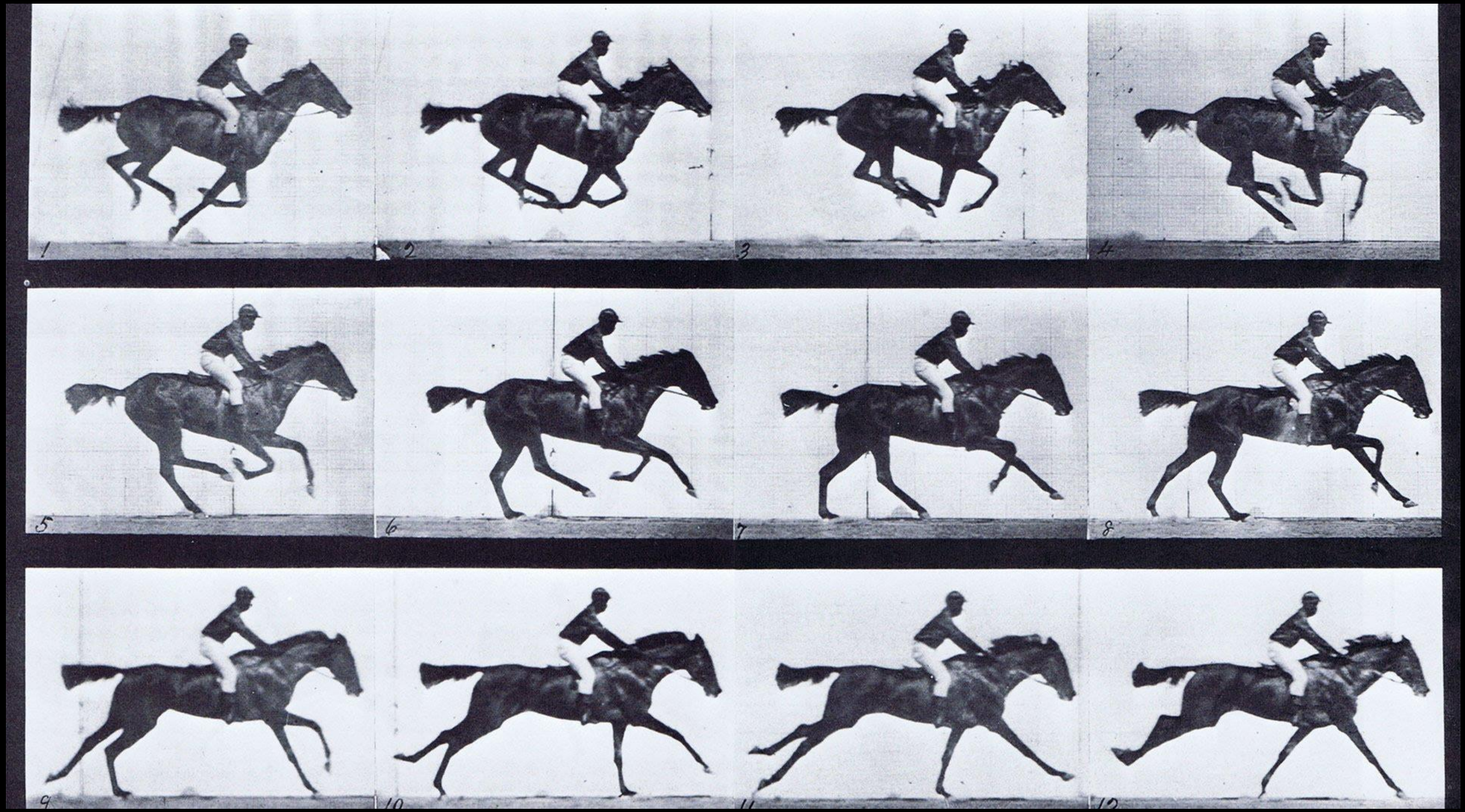
Data representation

Text



< feature size depends on which embedding we choose … but let’s say we used GloVe with 50-dimensional vectors.

Each word is represented as vector of 50 numbers.



*) The Horse in Motion, Eadweard Muybridge (1878); 7.4/10 - IMDb

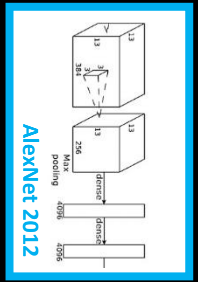
Data representation

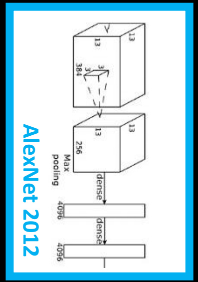
Video = sequence of frames

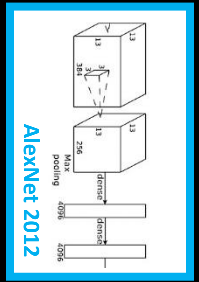
**) 1000 Frames of Hitchcock, Dave Pattern*

Video = sequence of frames

- Each frame can be described using a pre-trained Convolutional Neural Network





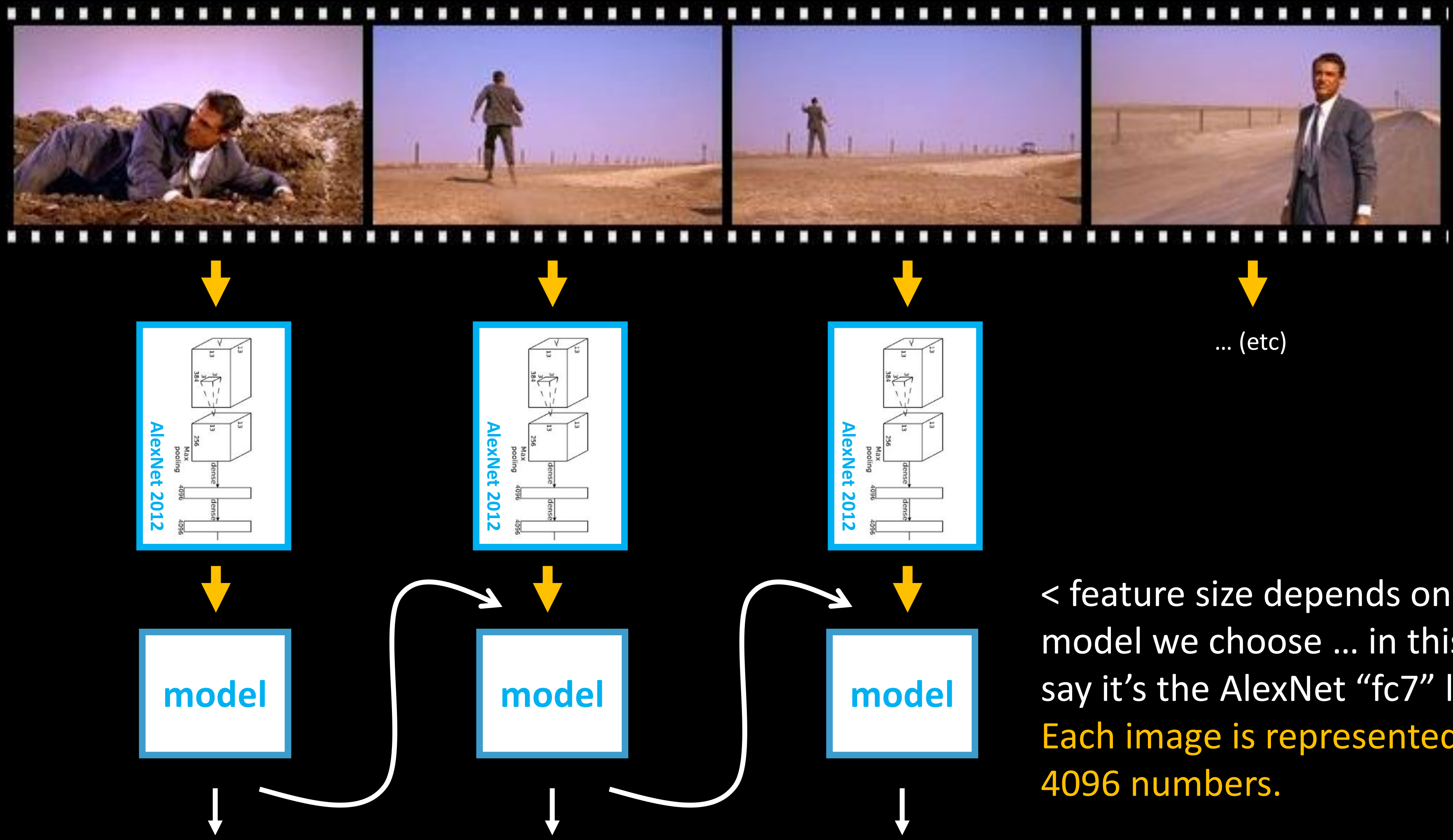


… (etc)

< feature size depends on which model we choose … in this case let’s say it’s the AlexNet “fc7” layer.
Each image is represented as vector of 4096 numbers.

Video = sequence of frames

- Each frame can be described using a pre-trained Convolutional Neural Network

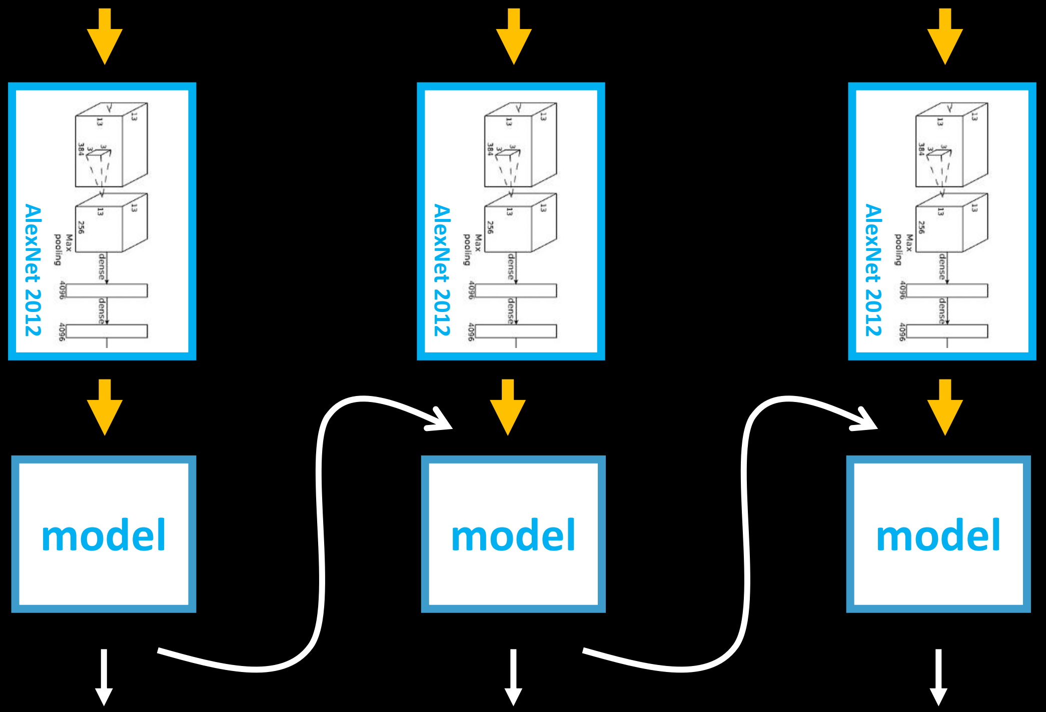


< feature size depends on which model we choose … in this case let's say it's the AlexNet "fc7" layer.
Each image is represented as vector of 4096 numbers.

Video = sequence of frames

- Each frame can be described using a pre-trained Convolutional Neural Network

*) Could also be pretty abstract …



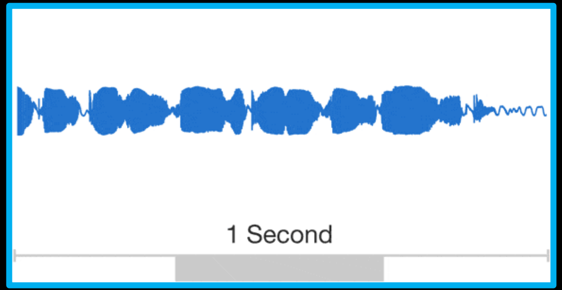
… (etc)

< feature size depends on which model we choose … in this case let's say it's the AlexNet "fc7" layer.
Each image is represented as vector of 4096 numbers.

Data representation

Audio

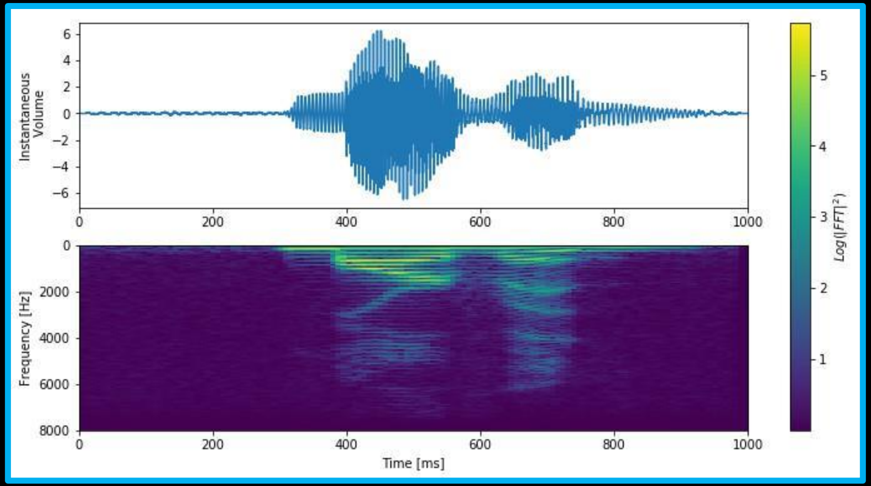
- Raw audio



Data representation

Audio

- Raw audio

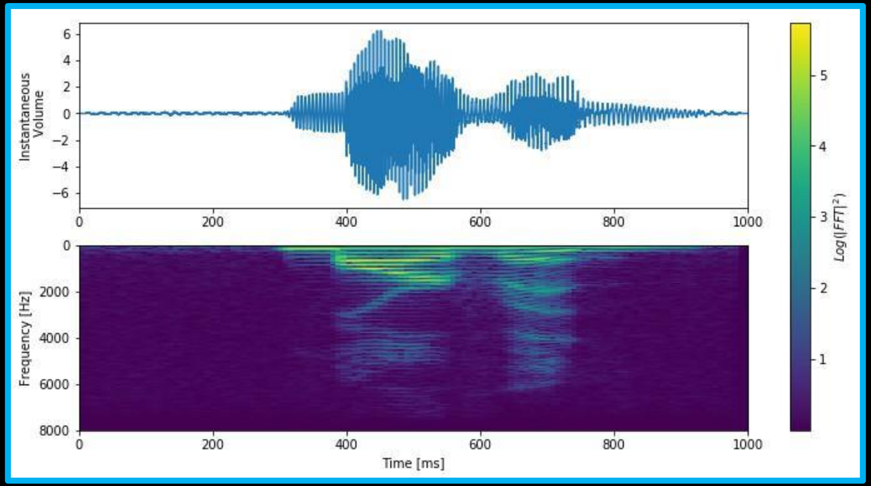


- Representation by spectrogram

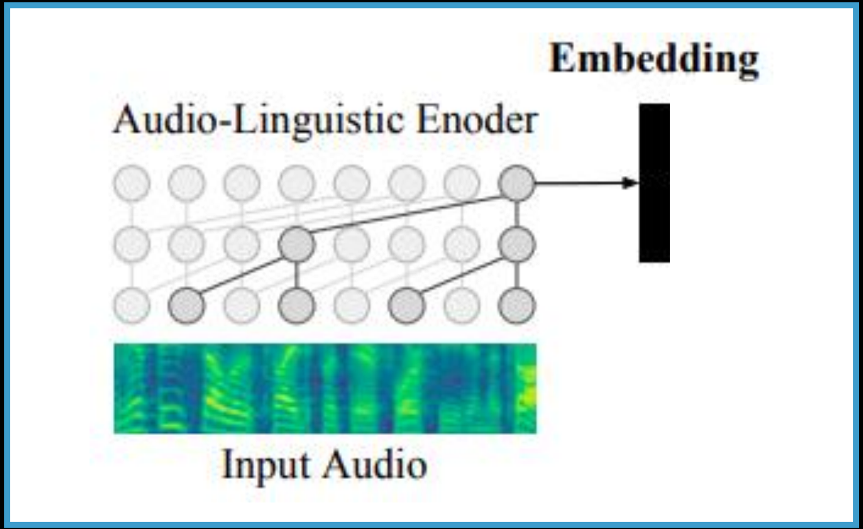
Data representation

Audio

- Raw audio



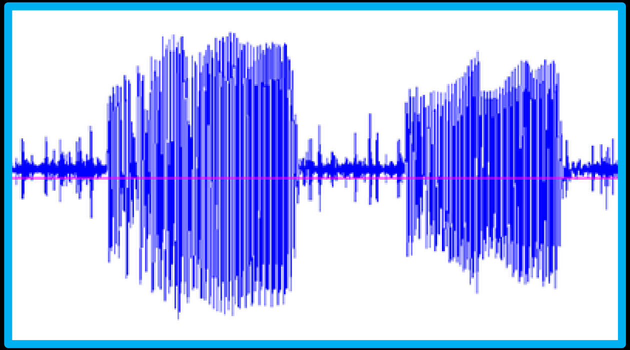
- Representation by spectrogram



- Embedding from a trained model

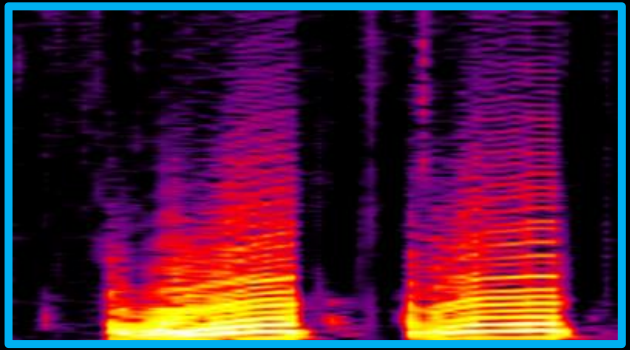
Spectrogram

original data



fft

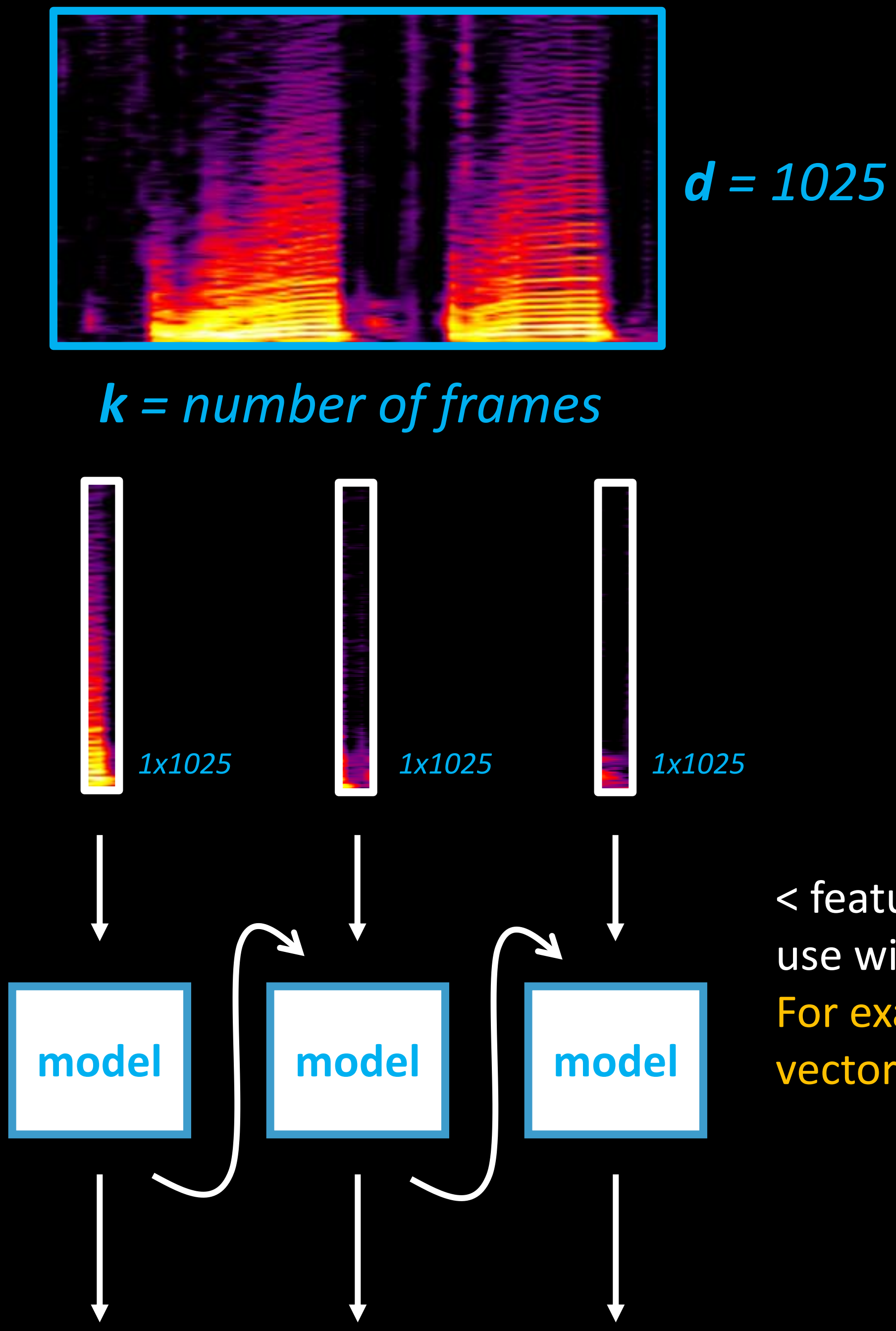
spectrograms



d *= 1025*

k *= number of frames*

- **Encode music using the Fourier Transform** (*fft*) to get **spectrogram** (which can be considered as image representation)
- We can **later decode the predictions** using the *inverse fft*



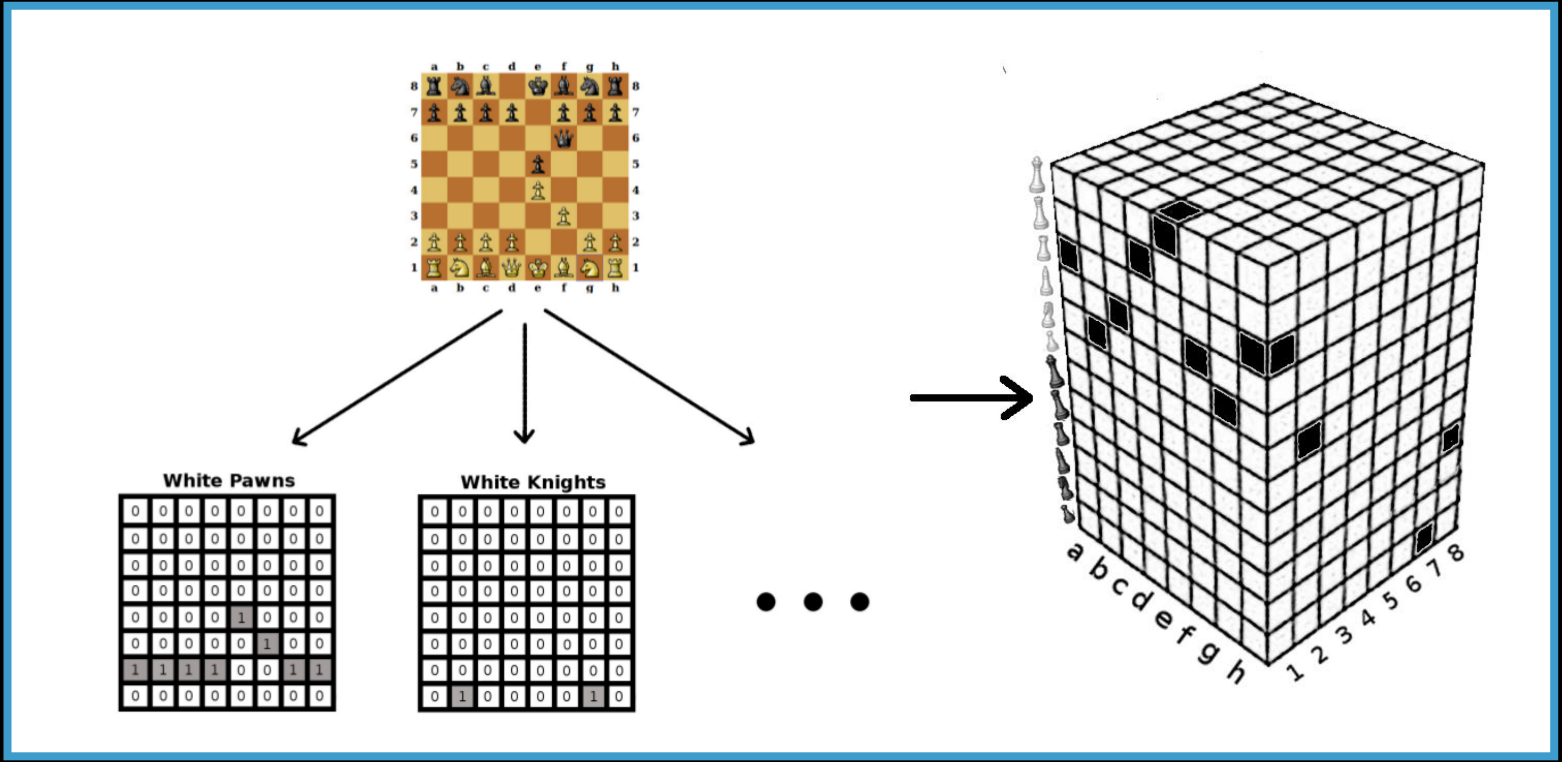
< feature size depends on settings we use with Fourier Transformation

For example each frame can be a vector of 1025 numbers.

Data representation

Actions

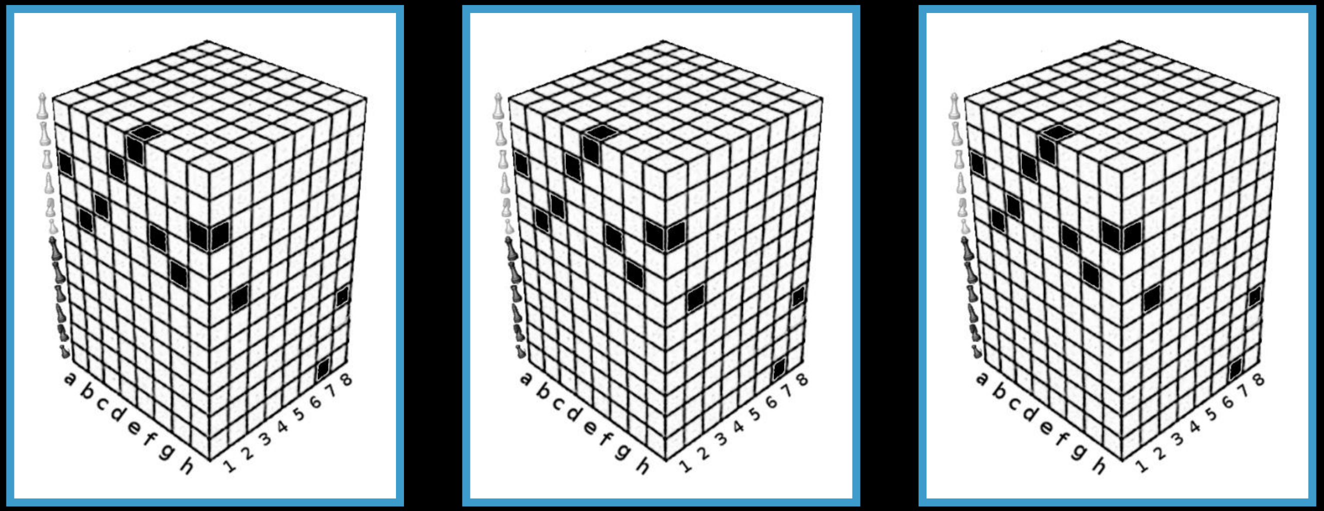
- Actions in a game (or sequences of game board states)



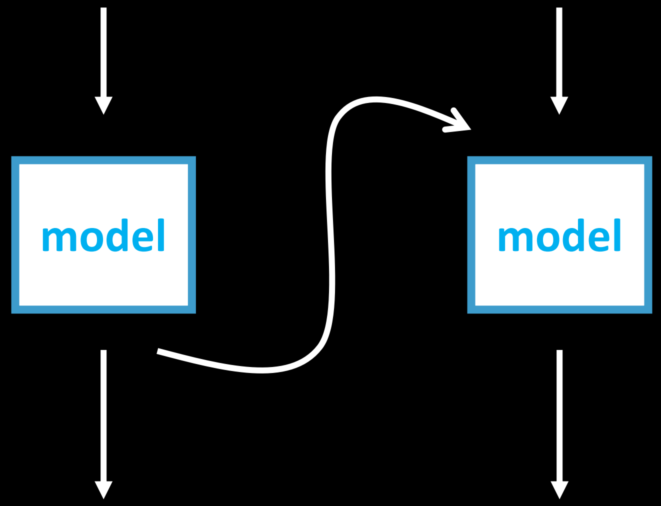
*) "Learning to Play Chess with Deep Reinforcement Learning" [link]

Data representation

Actions



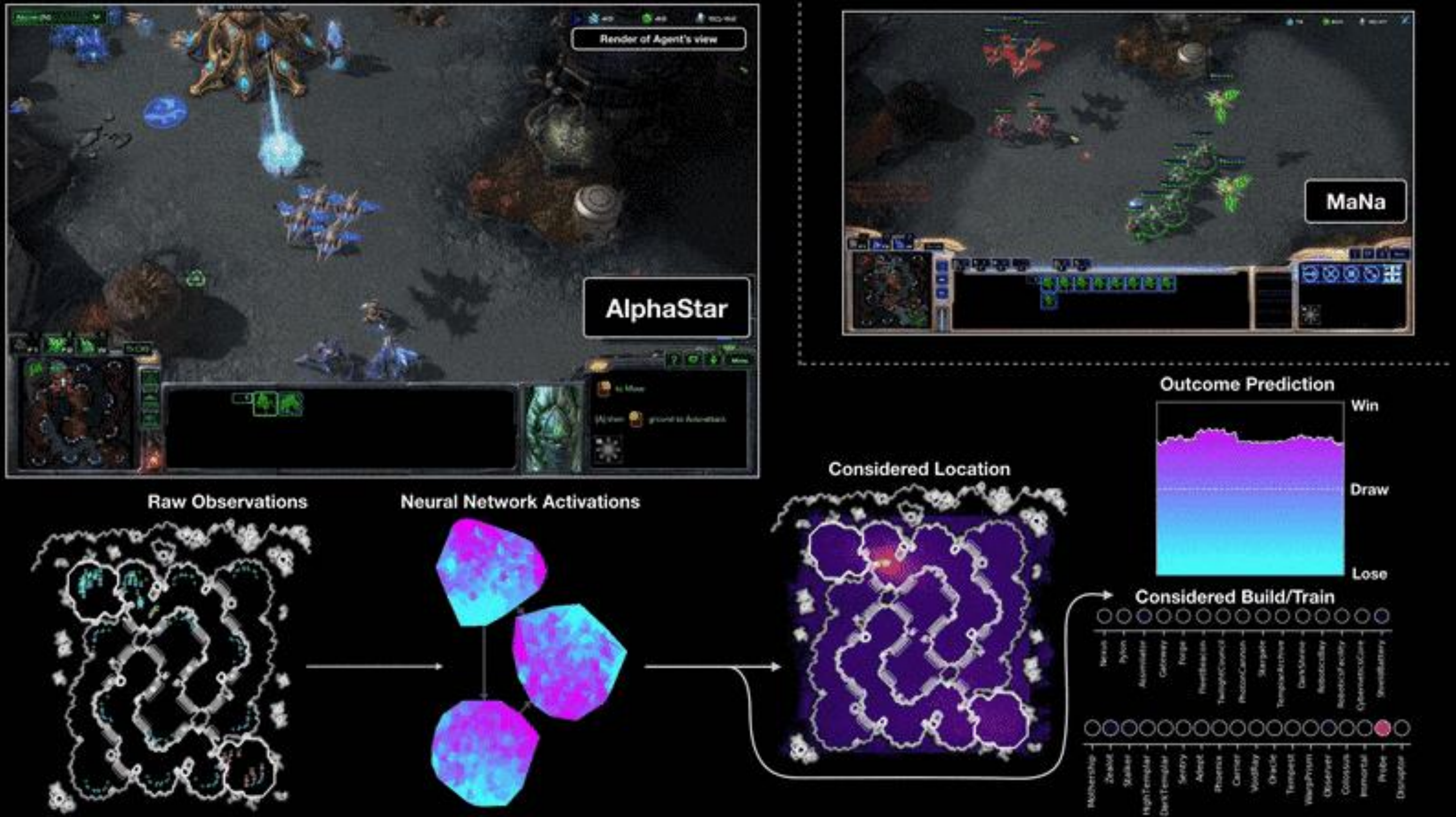
… (etc)



< feature size could be this 3D cube flattened into a single vector of numbers.

Data representation

Actions



***)** A bit more complicated task, usually uses Deep Reinforcement Learning.

*) “AlphaStar: Mastering the Real-Time Strategy Game StarCraft II” [link]

Sequential data

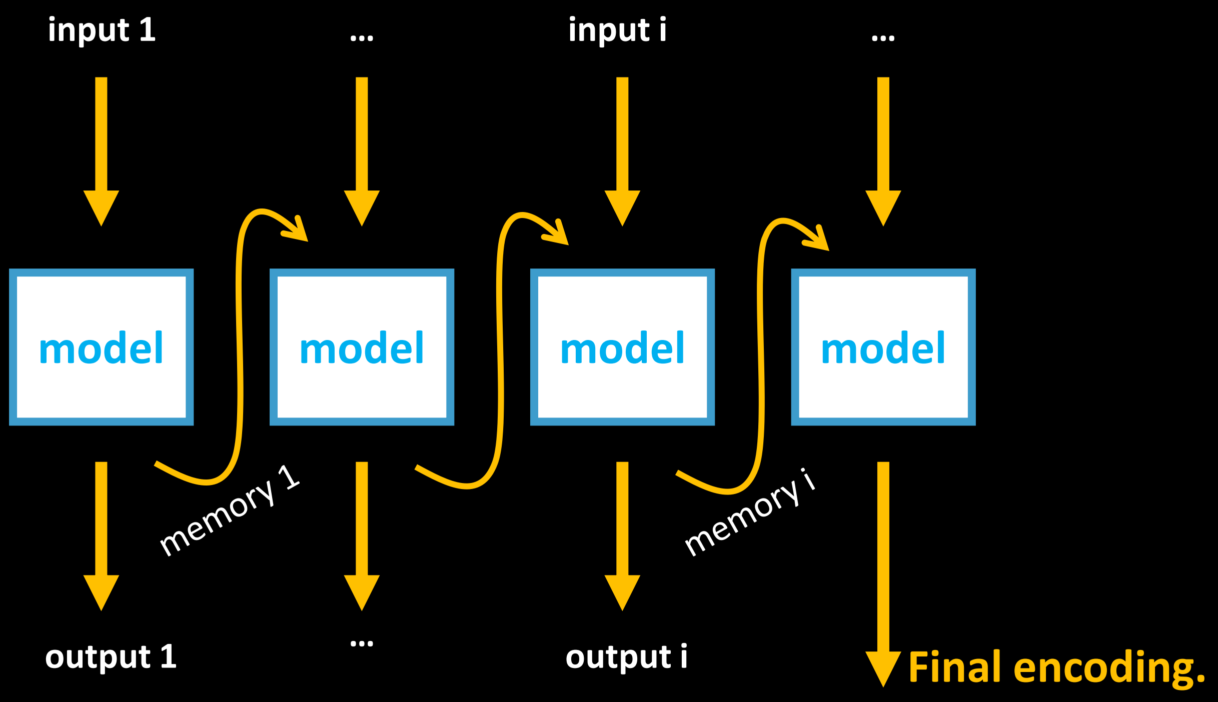
- We saw examples of:
  - Text
  - Video frames
  - Audio
  - Actions
  - *(and you can imagine other real-world data which we could abstract into representation of sequences)*

Sequential data

- We saw examples of:
  - Text
  - Video frames
  - Audio
  - Actions
  - *(and you can imagine other real-world data which we could abstract into representation of sequences)*
- … so far we used very simplified schematics of what the [**model**] is … let's explore this in more detail

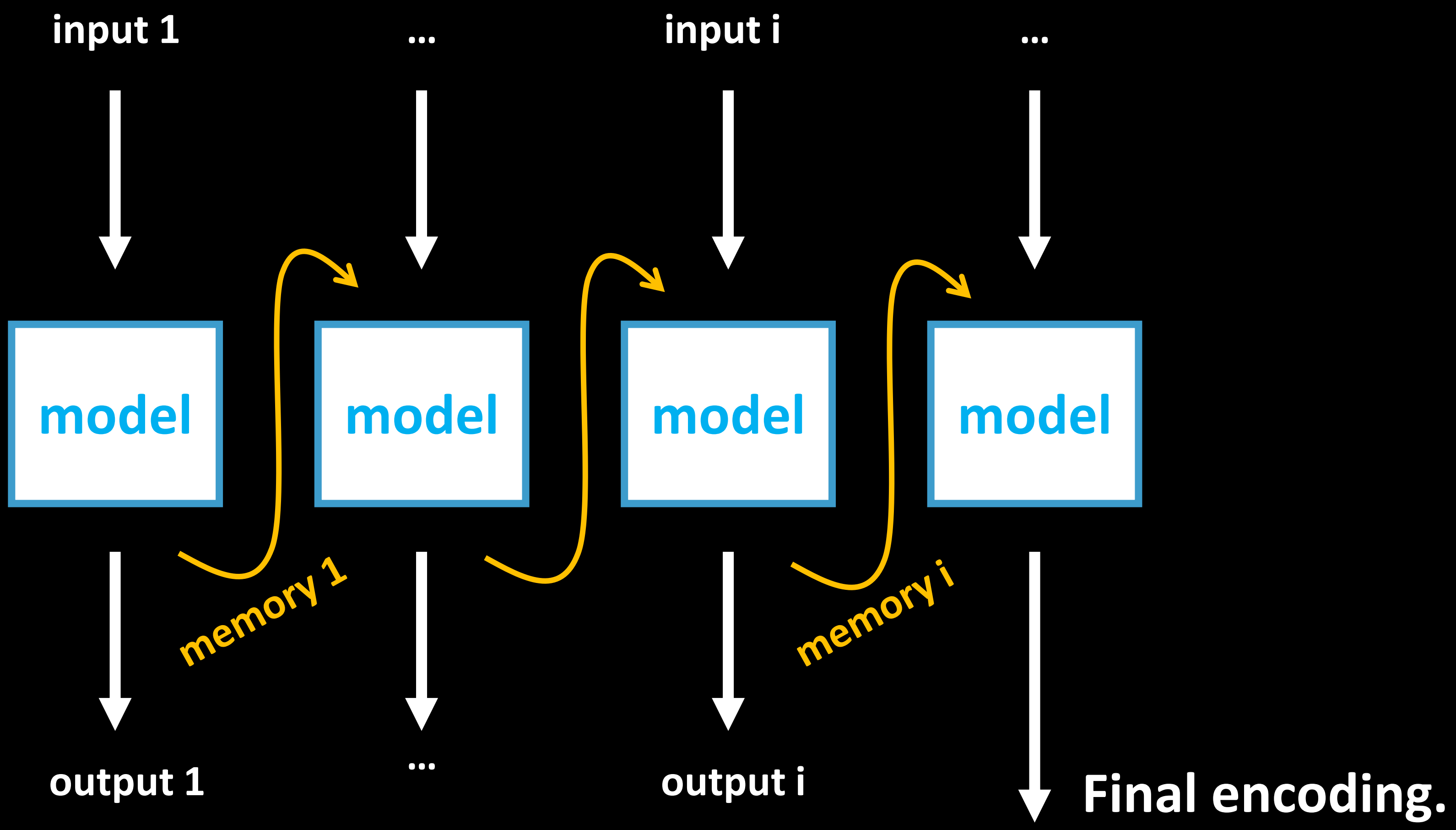
*) PS: now it’s a good time to take a pause

Model schematics



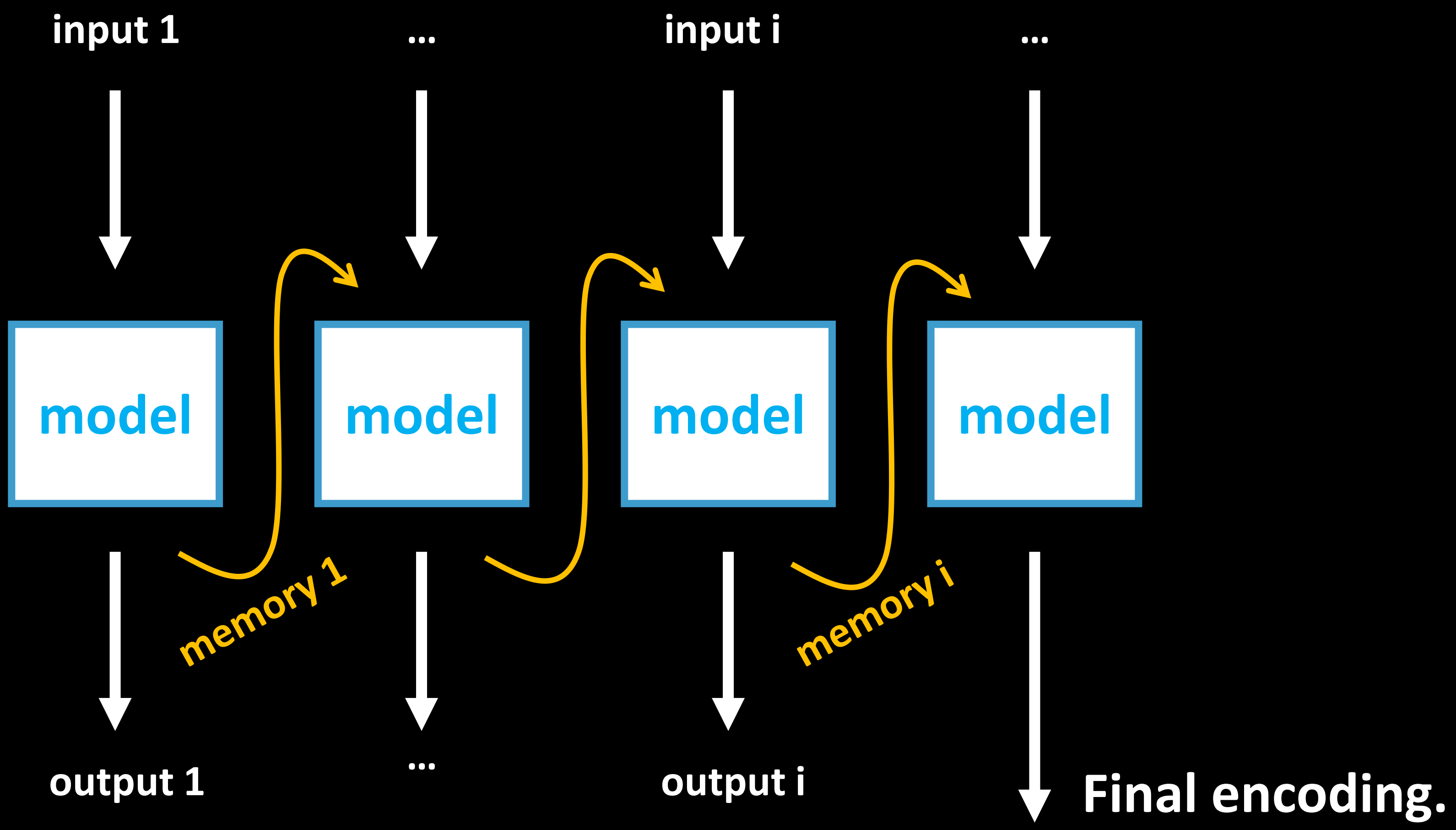
- What do we want from this model?

- What do we want?



- Learn to correctly assign the input-output label prediction
- Remember anything that is useful for the next prediction

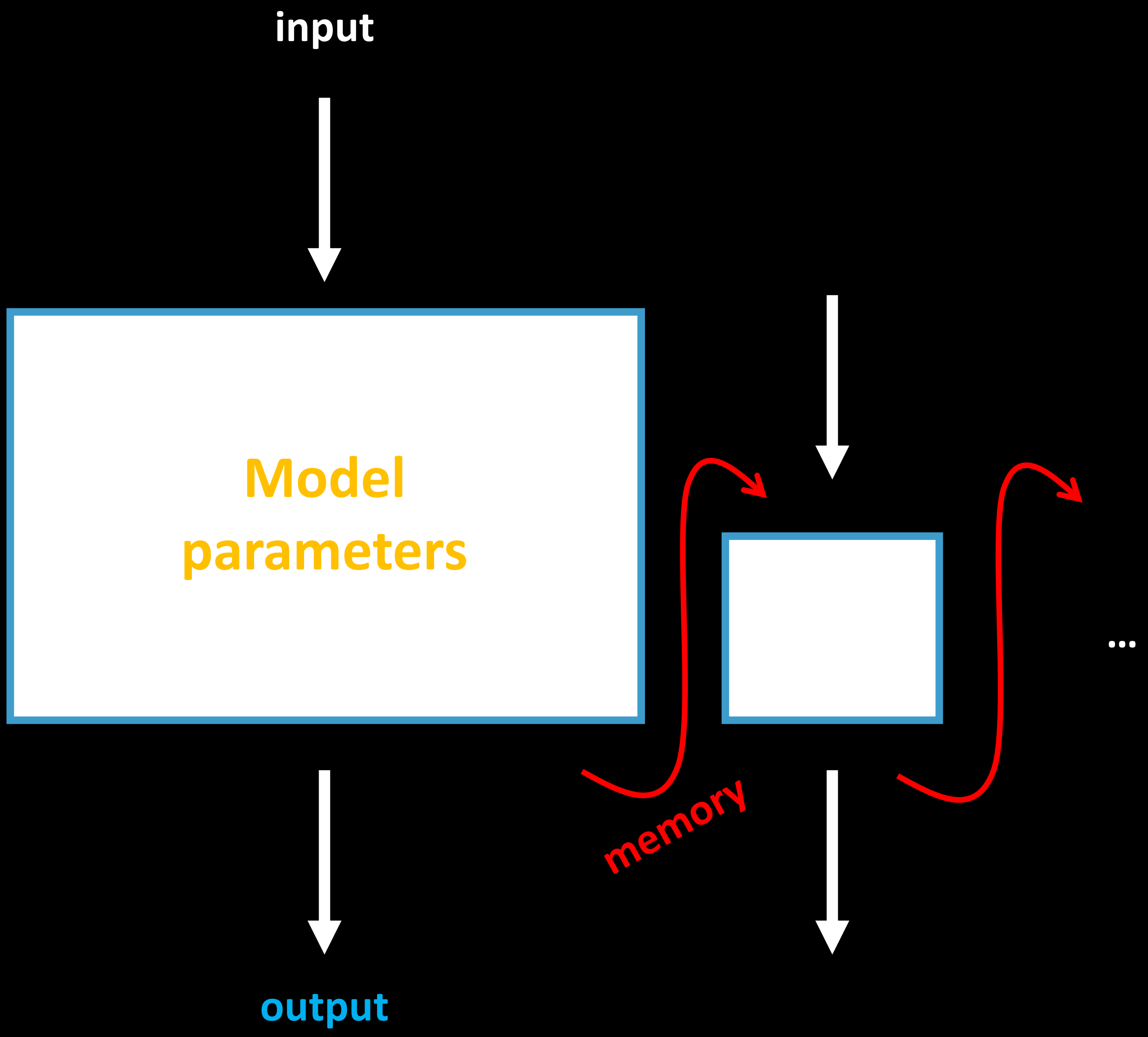
- What do we want?



- Learn to correctly assign the input-output label prediction
- Remember anything that is useful for the next prediction
- We get these properties by training …

Training

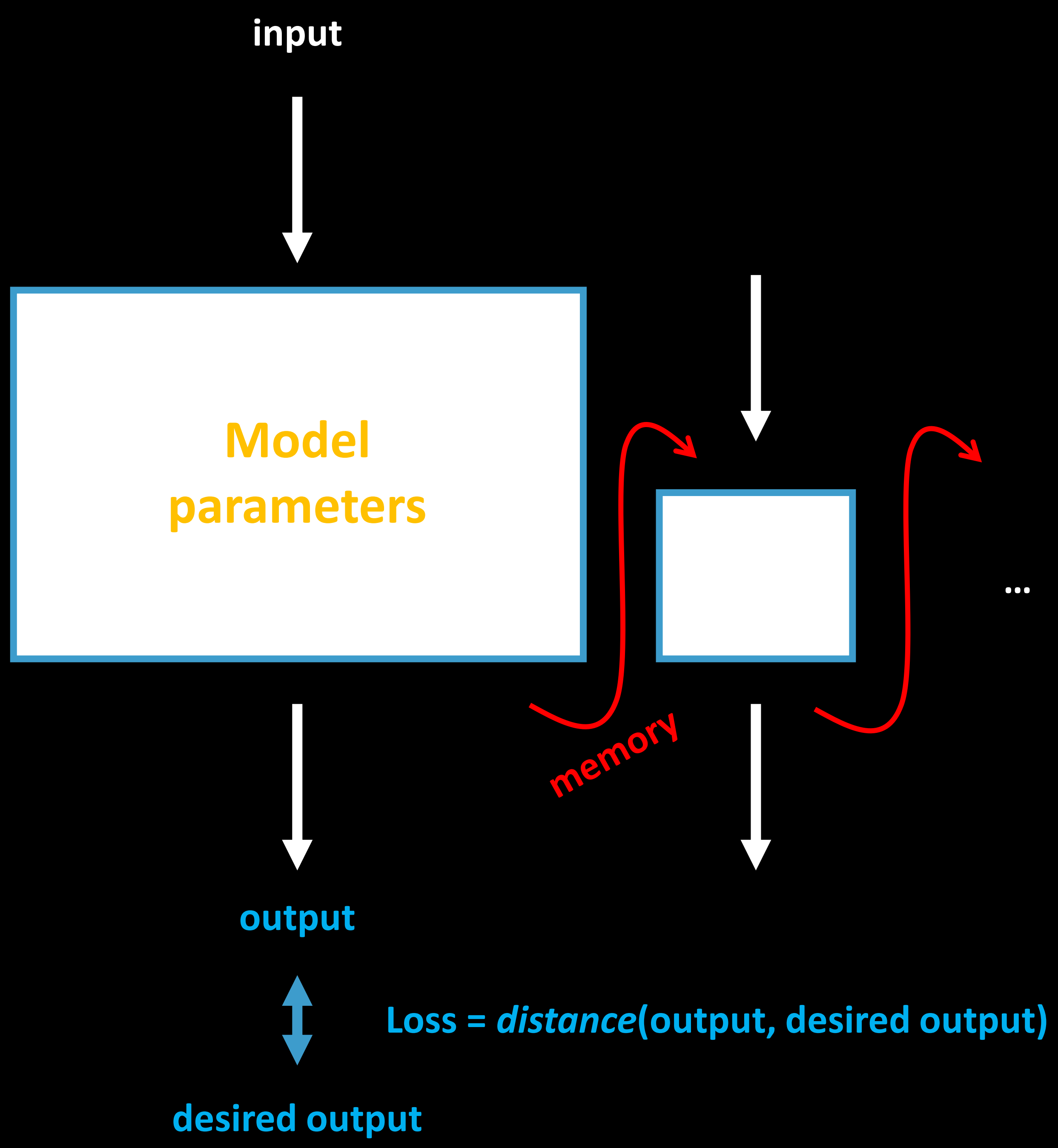
- We usually have some model **parameters** that we can set so that the model does what we want from it



Training

- We usually have some model **parameters** that we can set so that the model does what we want from it

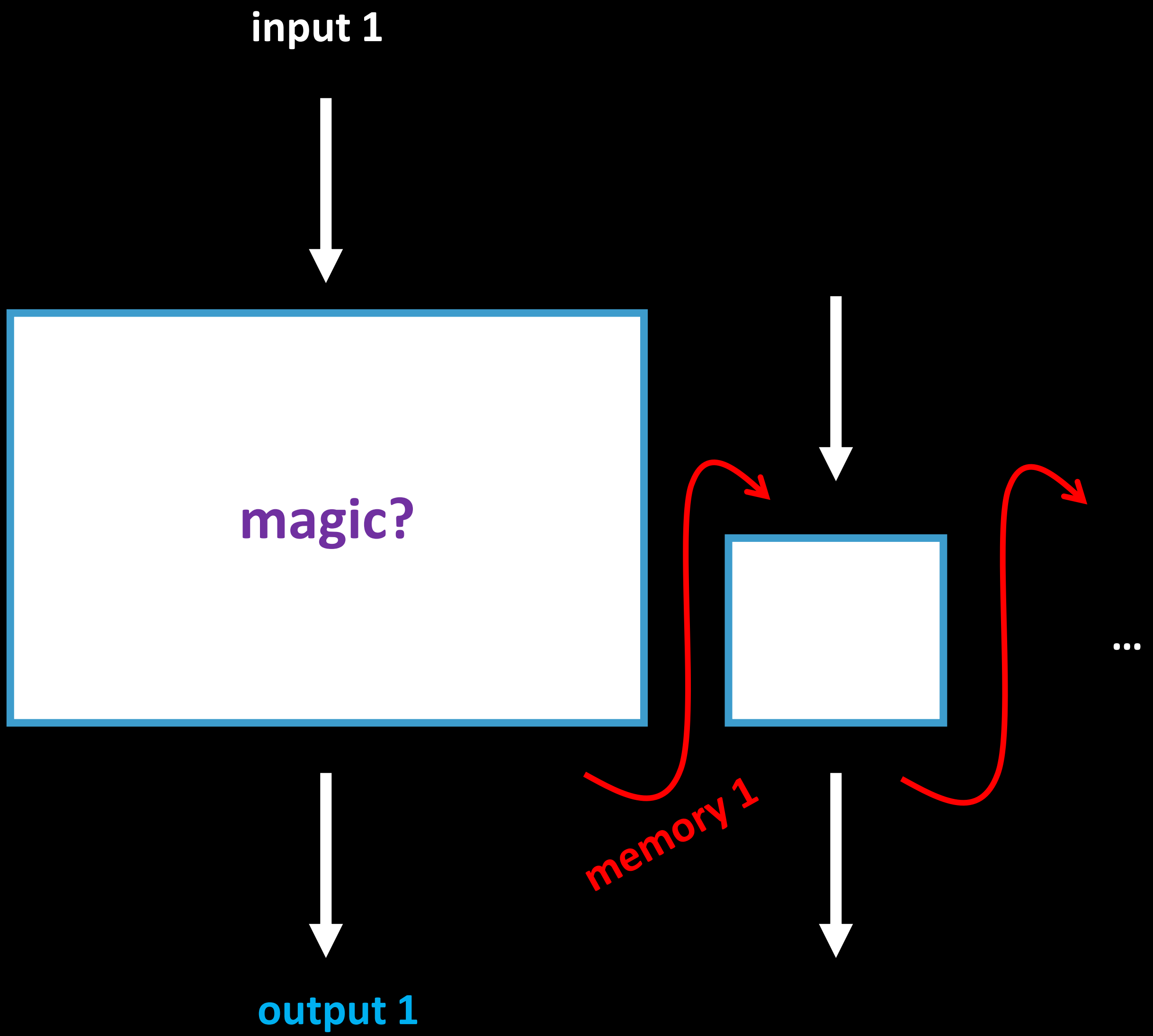
- We usually define what we want using a **loss function** on the predicted outputs and the desired output values.



Training

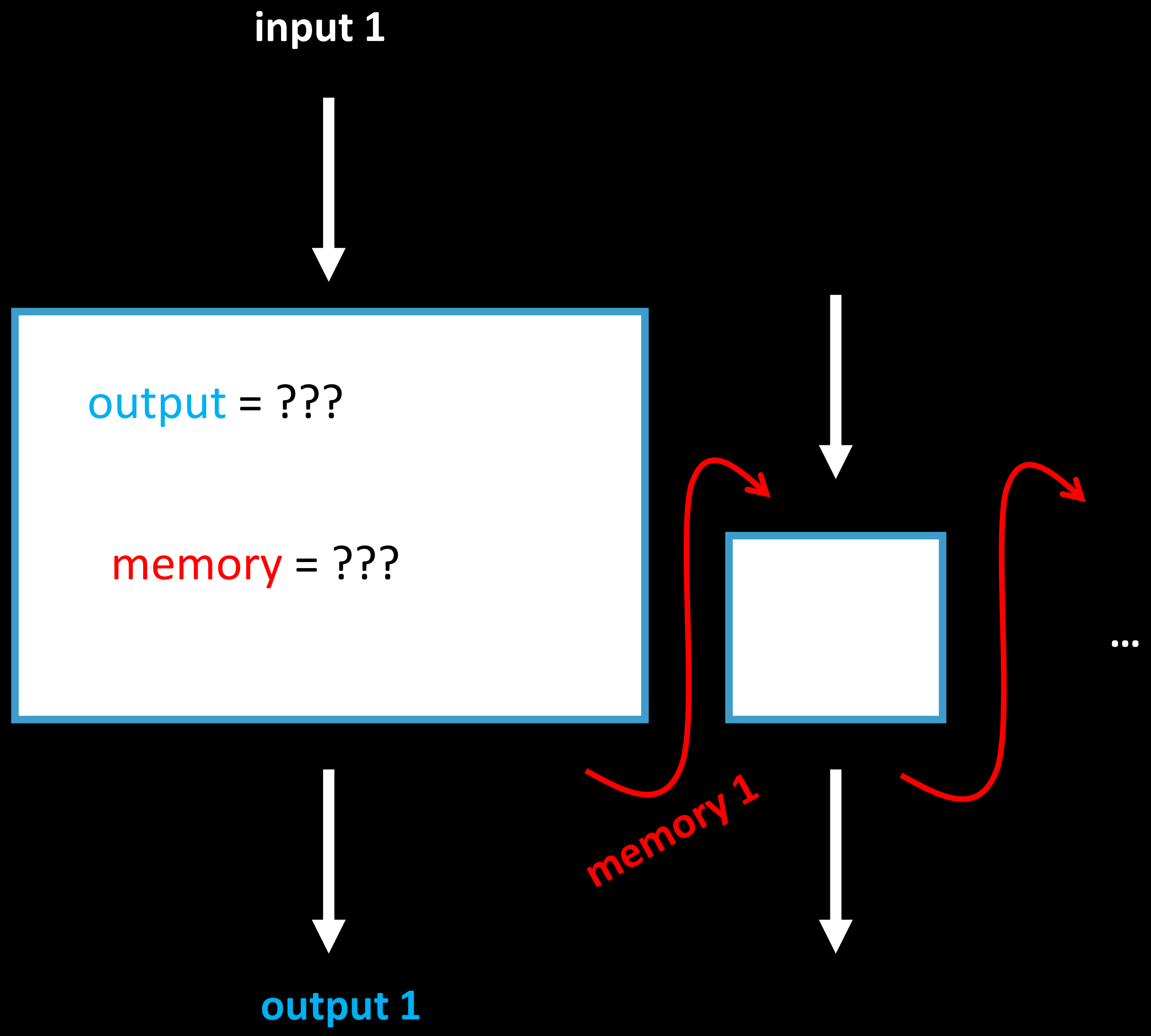
- Example (very simple model):

*) PS: this is going to be just a very arbitrary example of what is happening inside a model.
You don't need to calculate it by hand – its just to illustrate how we can influence information flow.



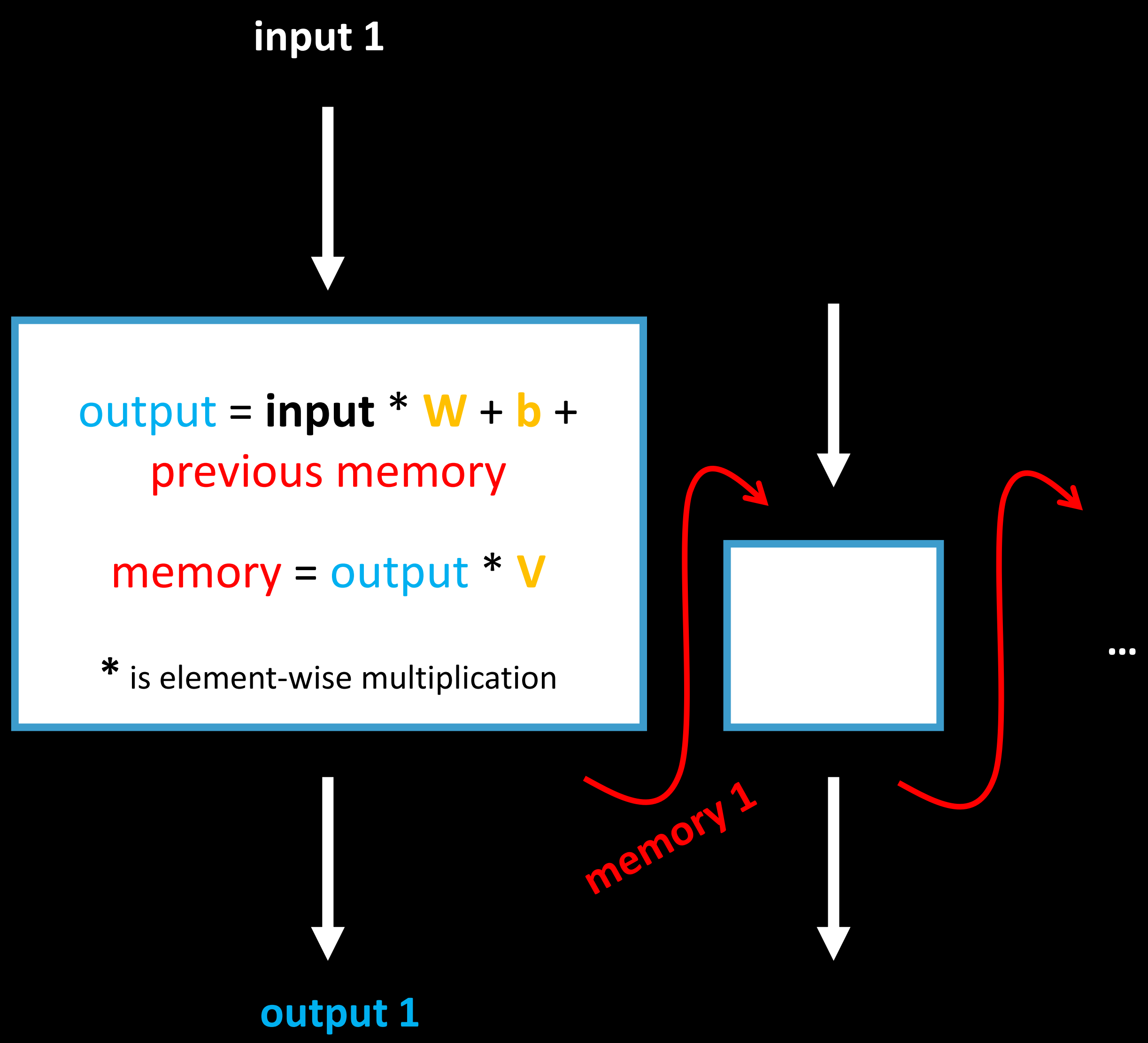
Training

- Example (very simple model):



Training

- Example (very simple model):



Training

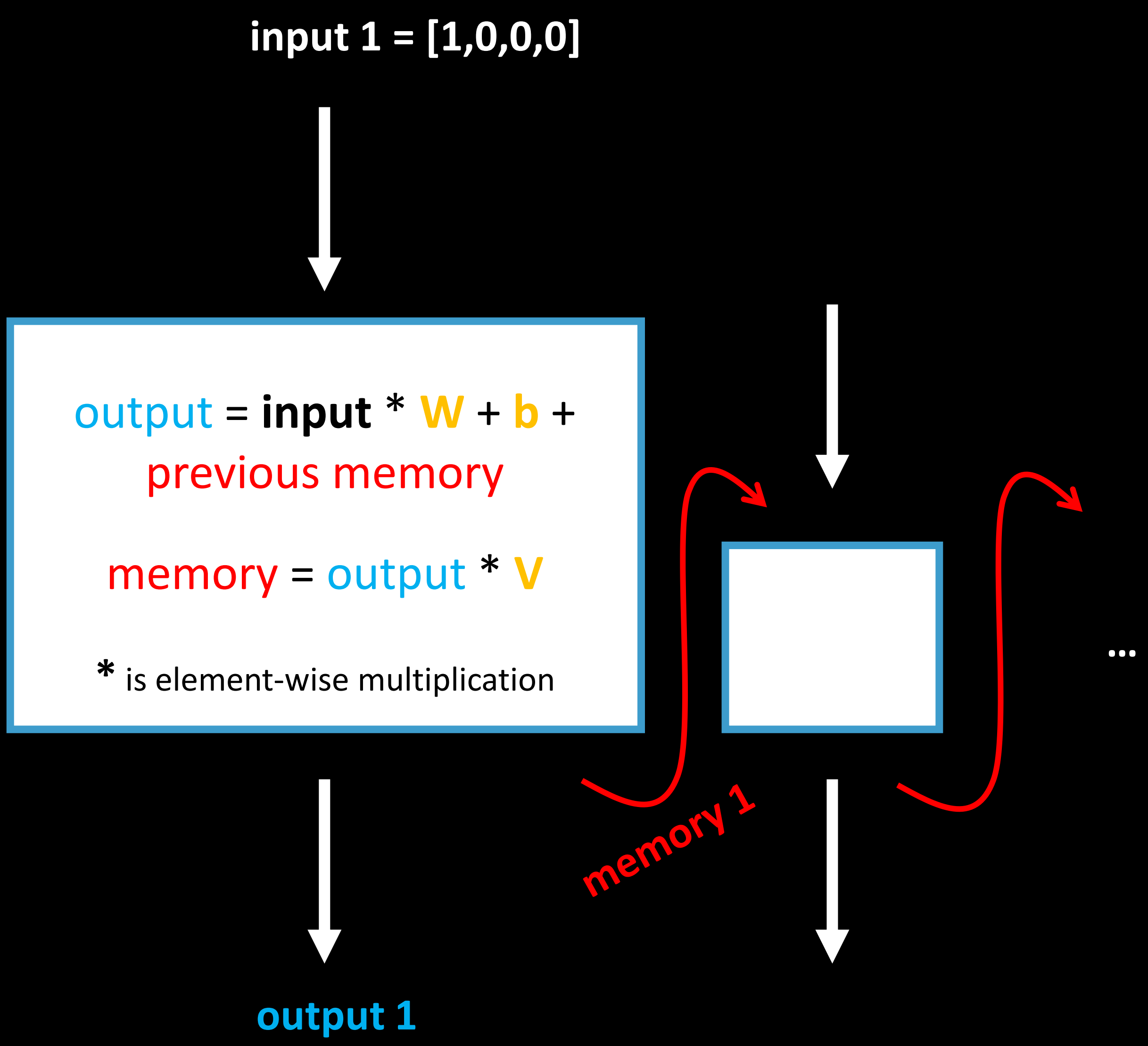
- Example (very simple model):

Let's say that someone told us that these values for the parameters are going to work the best:

W = [1,-1,0,0]

b = [0.1,0.1, 0.1, 0.1]

V = [1,1,-1,1]



Training

- Example (very simple model):

Let's say that someone told us that these values for the parameters are going to work the best:

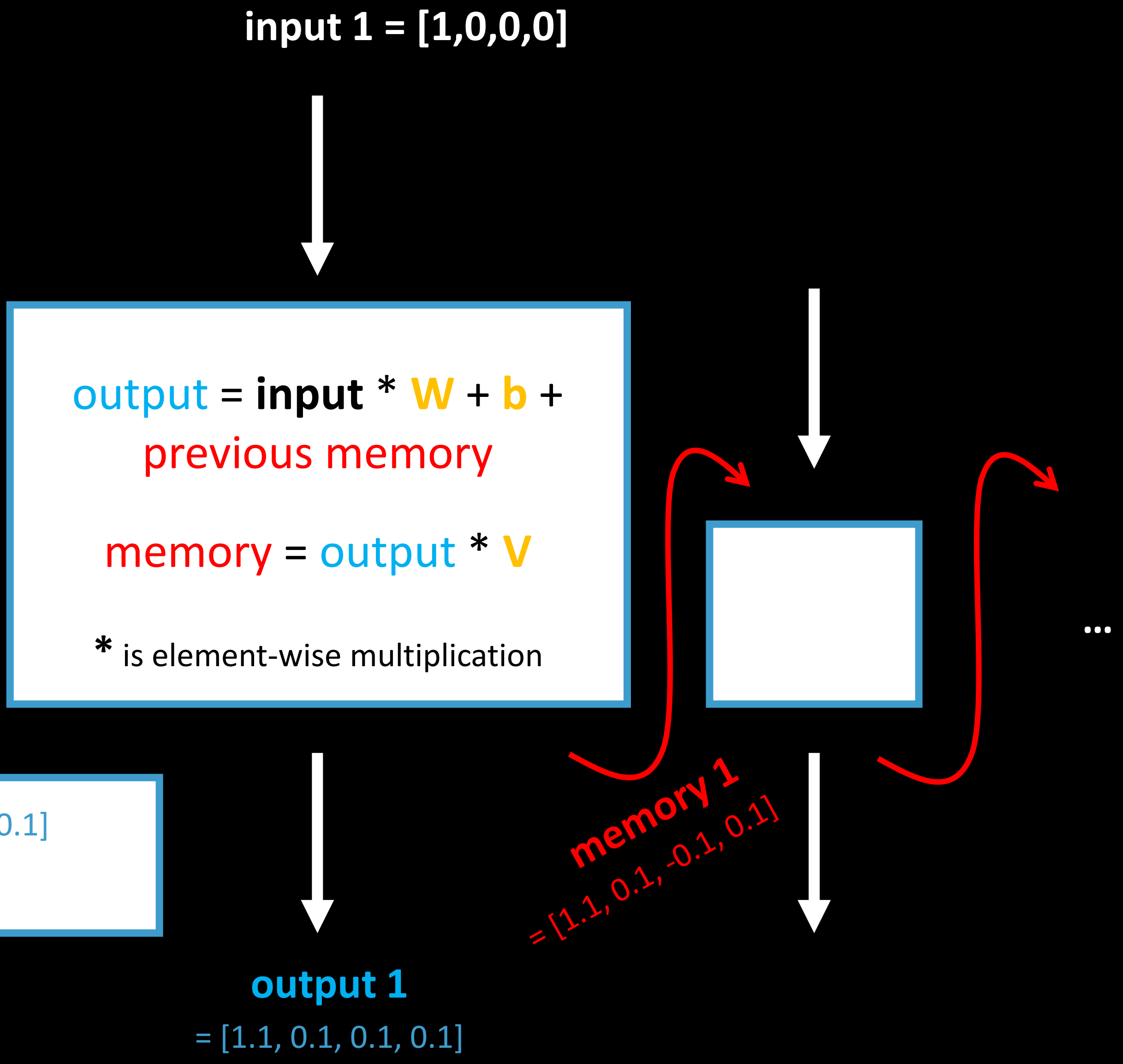
W = [1,-1,0,0]

b = [0.1,0.1, 0.1, 0.1]

V = [1,1,-1,1]

output 1 = **[1,0,0,0]** * [1,-1,0,0] + [0.1,0.1, 0.1, 0.1] + [0,0,0,0] = [1.1, 0.1, 0.1, 0.1]

memory 1 = [1.1, 0.1, 0.1, 0.1] * [1,1,-1,1] = [1.1, 0.1, -0.1, 0.1]



Training

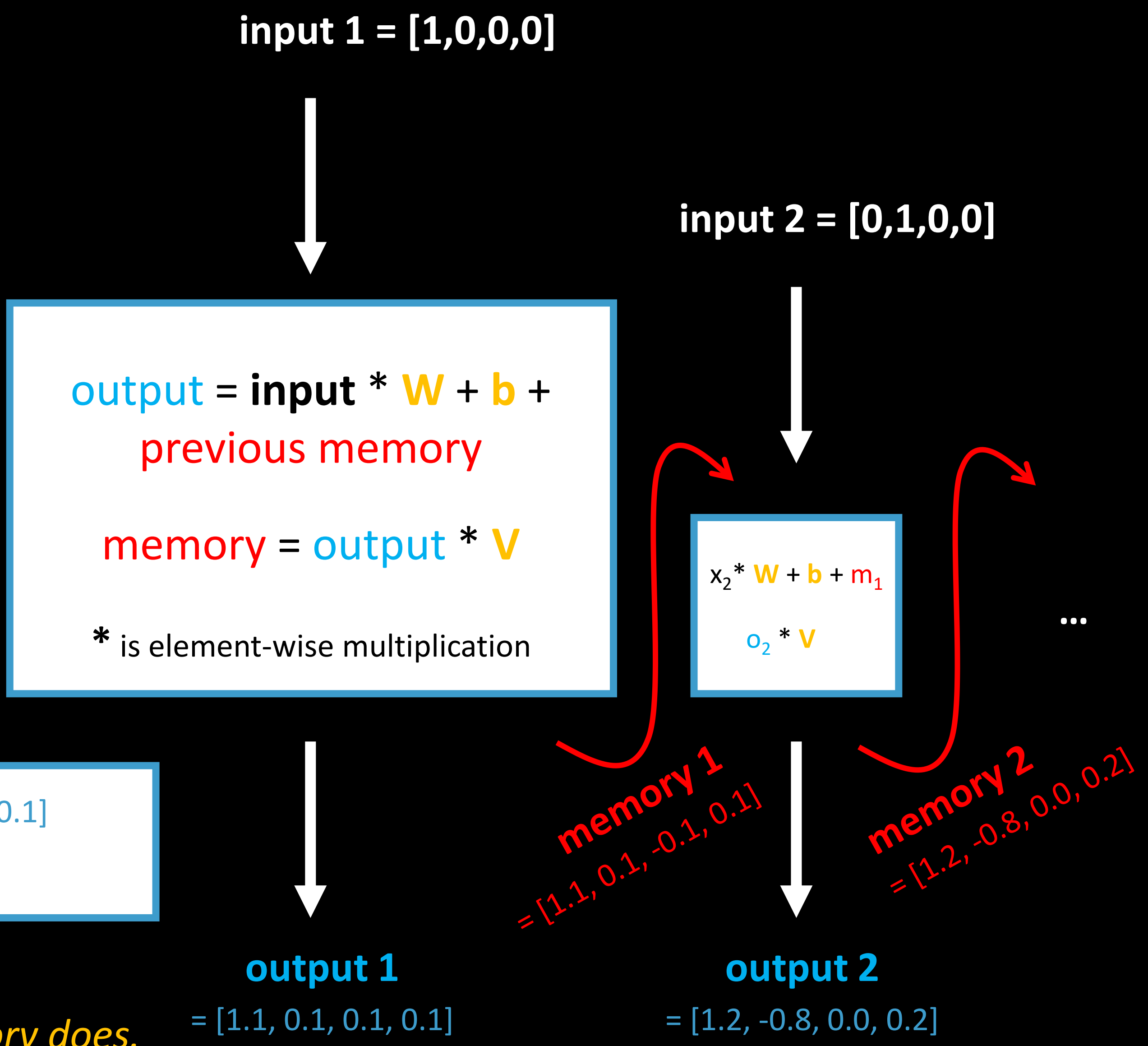
- Example (very simple model):

Let's say that someone told us that these values for the parameters are going to work the best:

W = [1,-1,0,0]

b = [0.1,0.1, 0.1, 0.1]

V = [1,1,-1,1]



output 1 = **[1,0,0,0]** * [1,-1,0,0] + [0.1,0.1, 0.1, 0.1] + [0,0,0,0] = [1.1, 0.1, 0.1, 0.1]

memory 1 = [1.1, 0.1, 0.1, 0.1] * [1,1,-1,1] = [1.1, 0.1, -0.1, 0.1]

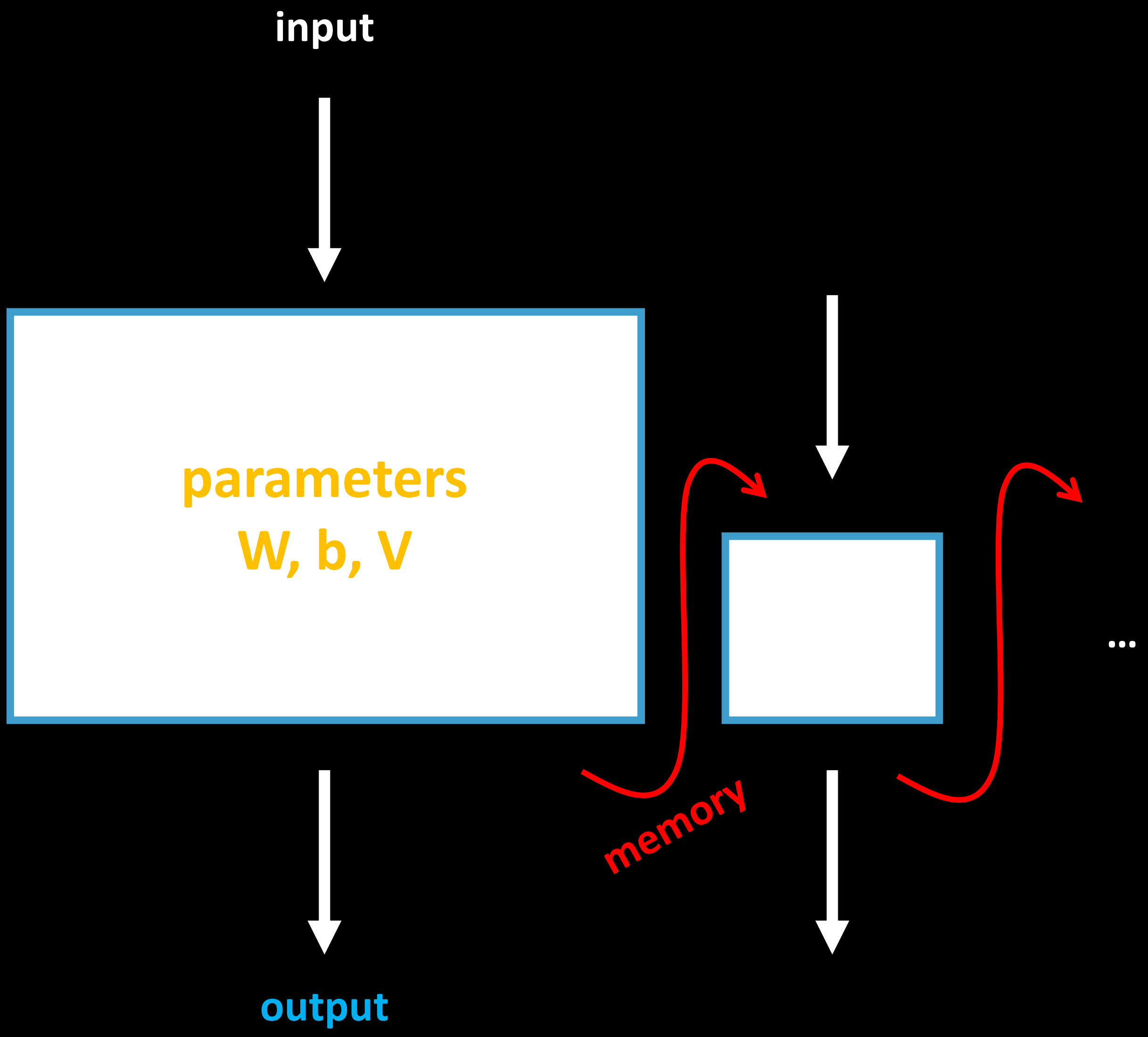
The parameter values don't change, but the input and memory does.

output 2 = **[0,1,0,0]** * [1,-1,0,0] + [0.1,0.1, 0.1, 0.1] + [1.1, 0.1, -0.1, 0.1] = [1.2, -0.8, 0.0, 0.2]

memory 2 = [1.2, -0.8, 0.0, 0.2] * [1,1,-1,1] = [1.2, -0.8, 0.0, 0.2]

Training

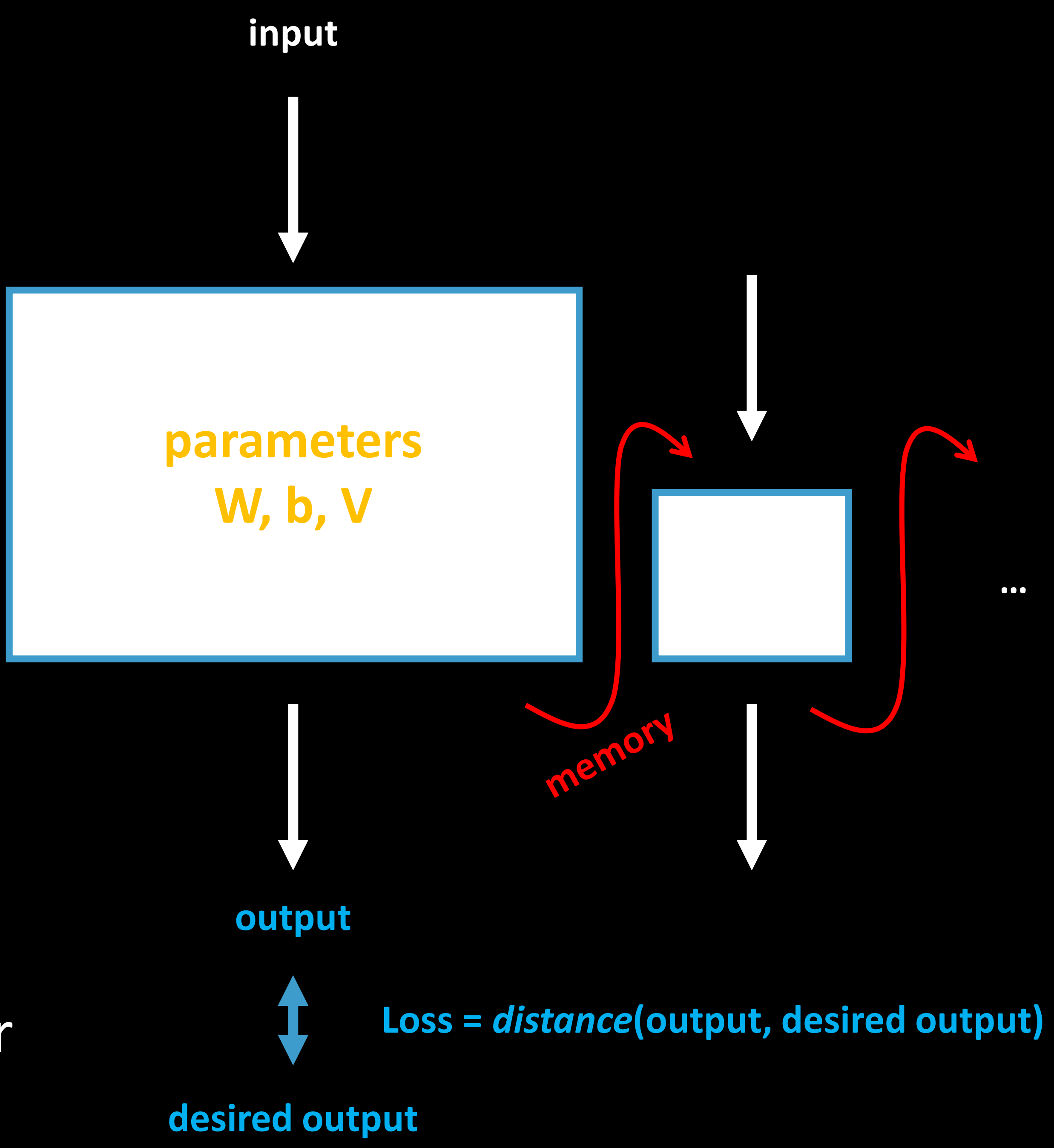
- This is to illustrate that by using **parameters** to influence what happens to the input (W, b) and what is kept in the memory (V) …
- … we can influence the behaviour of the **model**



Training

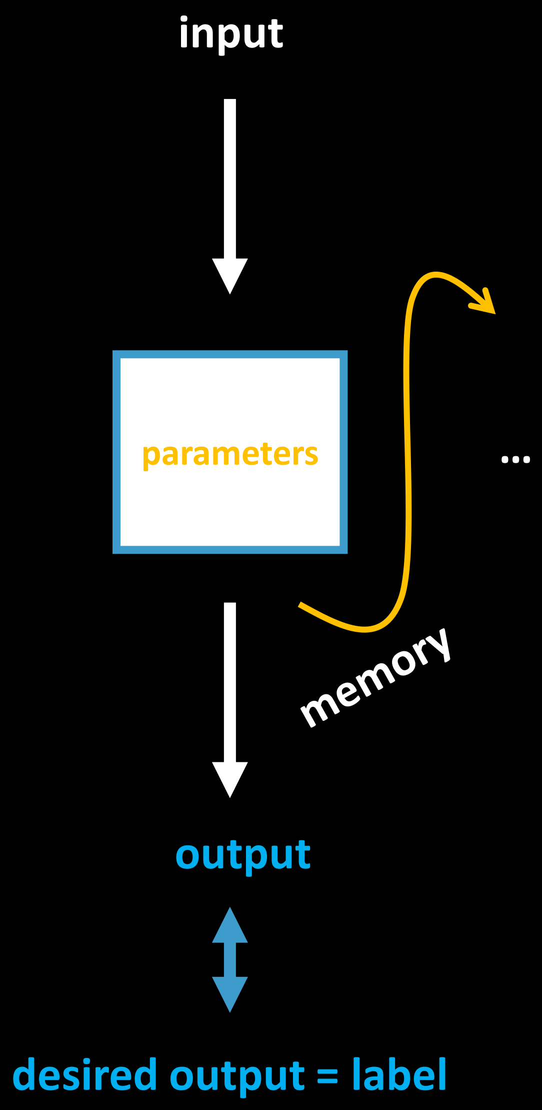
- This is to illustrate that by using **parameters** to influence what happens to the input (W, b) and what is kept in the memory (V) …
- … we can influence the behaviour of the **model**

- Task: Iteratively change parameters (**W, b, V**) so that the **loss** gets smaller

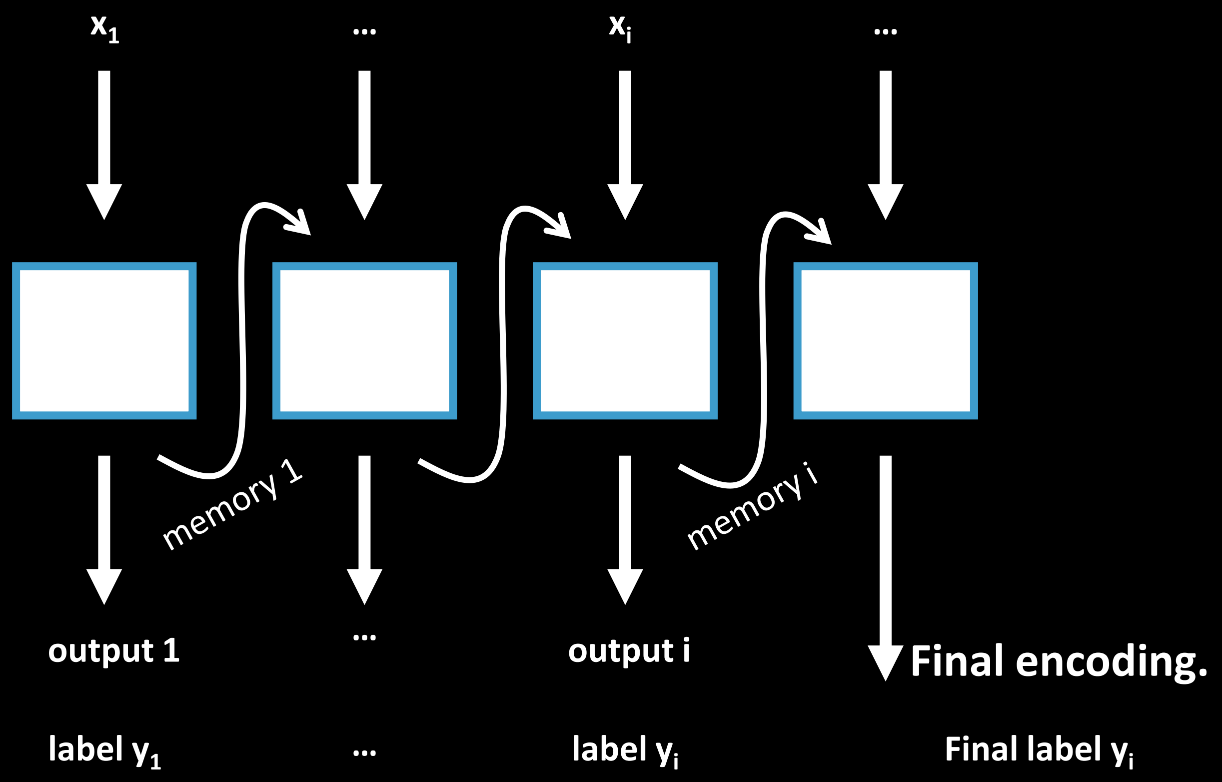


Training

- Find **parameters** which will minimize the distance between the prediction our model is giving and the labels we have (**loss**)



Task: classification / regression / generation



- Different scenarios for sequential modelling – depending on the task and the dataset we might be using different model variants

Task: classification / regression / generation

- Classification
  - We want to classify the input data into its class.
  - For example: input = mail text, label = spam / not-spam

Task: classification / regression / generation

- Classification
  - We want to classify the input data into its class.
  - For example: input = mail text, label = spam / not-spam

- Regression
  - We want to assign a continuous value to the input data.
  - For example: input = movie frames, label = expected IMDb rating

**) ps: classification and regression is very similar.*

Task: classification / regression / generation

- Classification
  - We want to classify the input data into its class.
  - For example: input = mail text, label = spam / not-spam

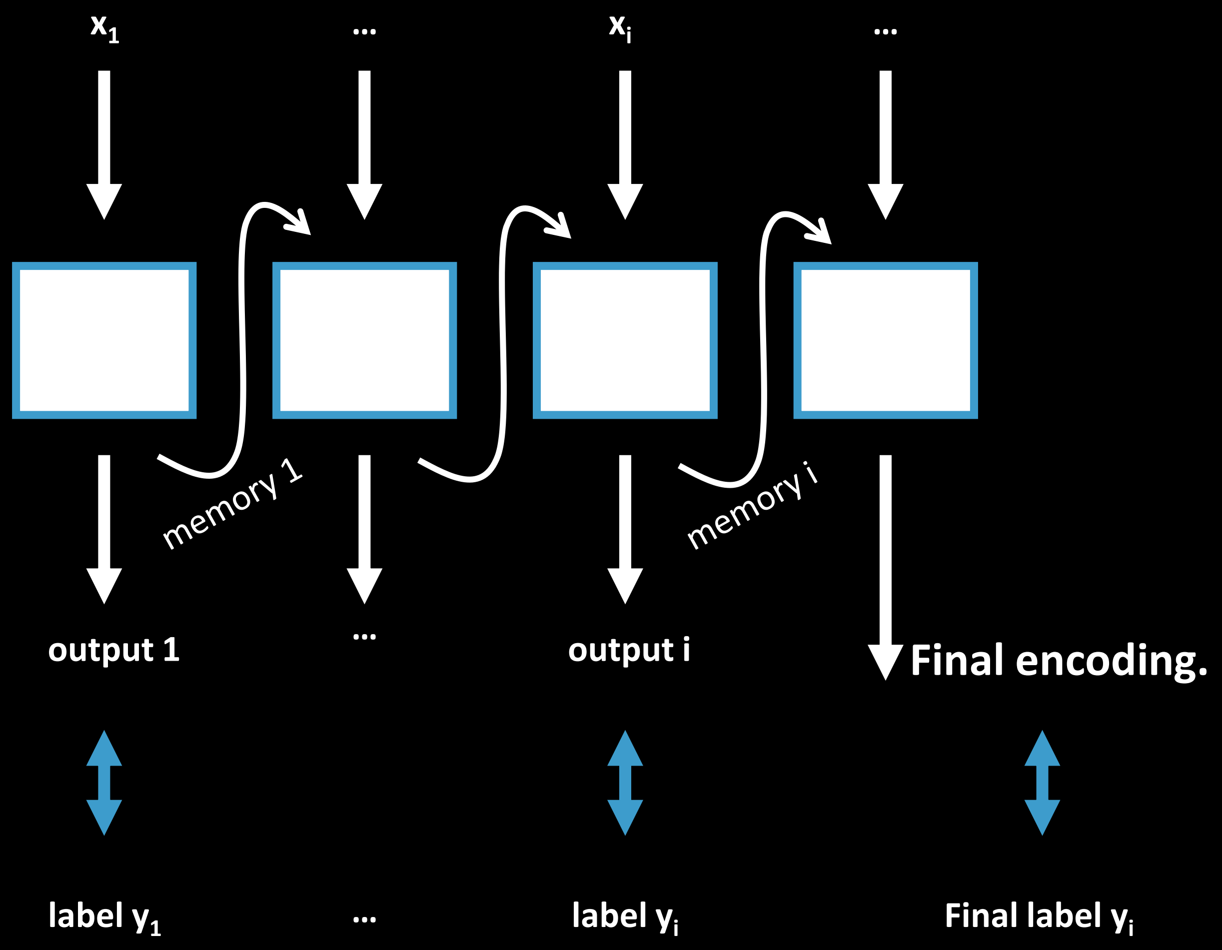
- Regression
  - We want to assign a continuous value to the input data.
  - For example: input = movie frames, label = expected IMDb rating

**) ps: classification and regression is very similar.*

- Generation
  - We want to use the model to predict a believable continuation to what we show it. *(* more in the next class)*

- (1) We might have an expected label for each output ("many-to-many" type of prediction):

Input data: x_i, y_i



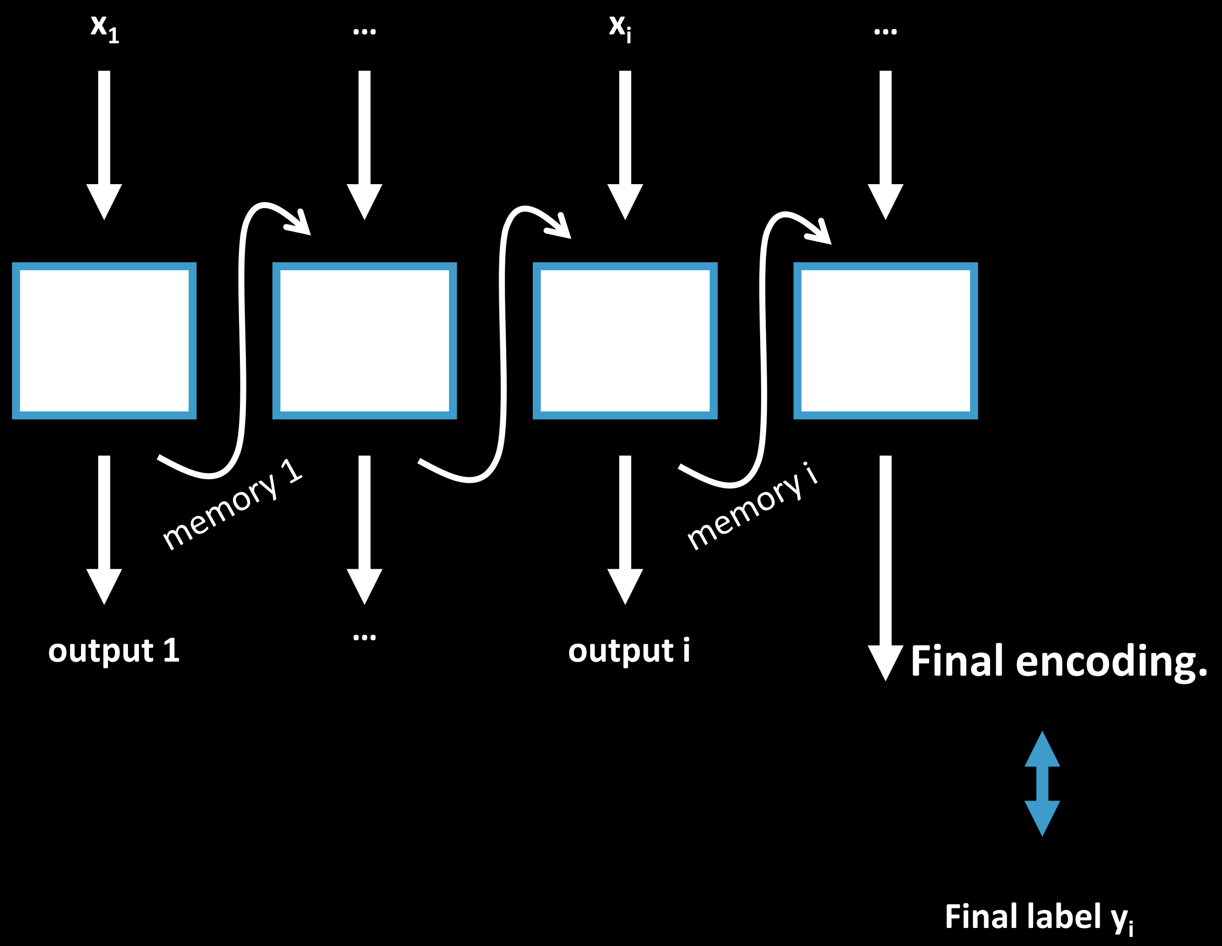
(1) Many-to-many,
can be classification, regression or even generation.

< Loss can include all of these distances

- (2) We might have a label only for the whole sequence (typically our x is made of individual words of some document and we have a single label y describing the whole document):

(2) Many-to-one,
can be classification, regression.

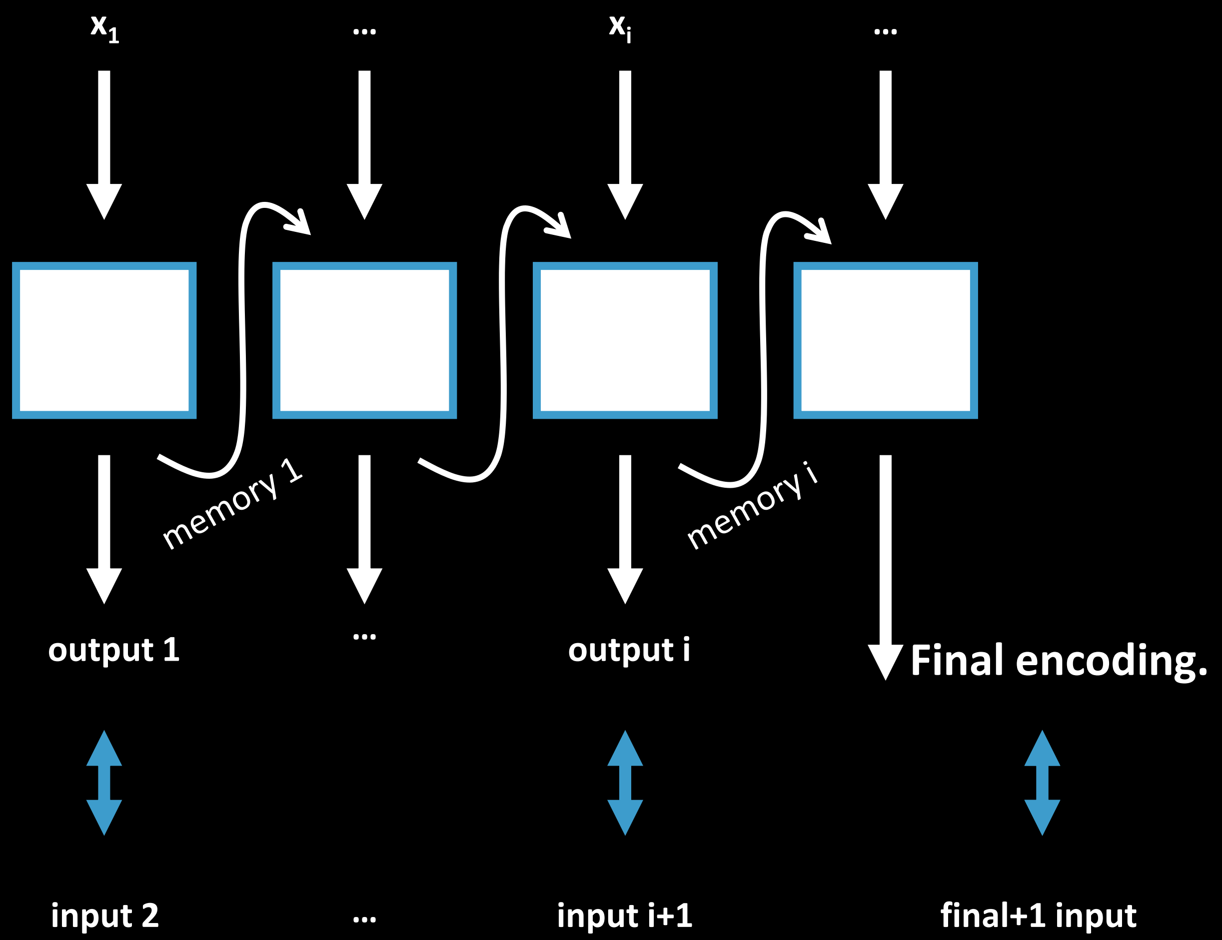
Input data: x_i, y



< Loss will then look only at the distance between the label and the final encoding

- (3) Finally we might want to generate data with this model *(spoilers for the next class)* – then we would have labels corresponding to the next item in the sequence:

Input data: x_i

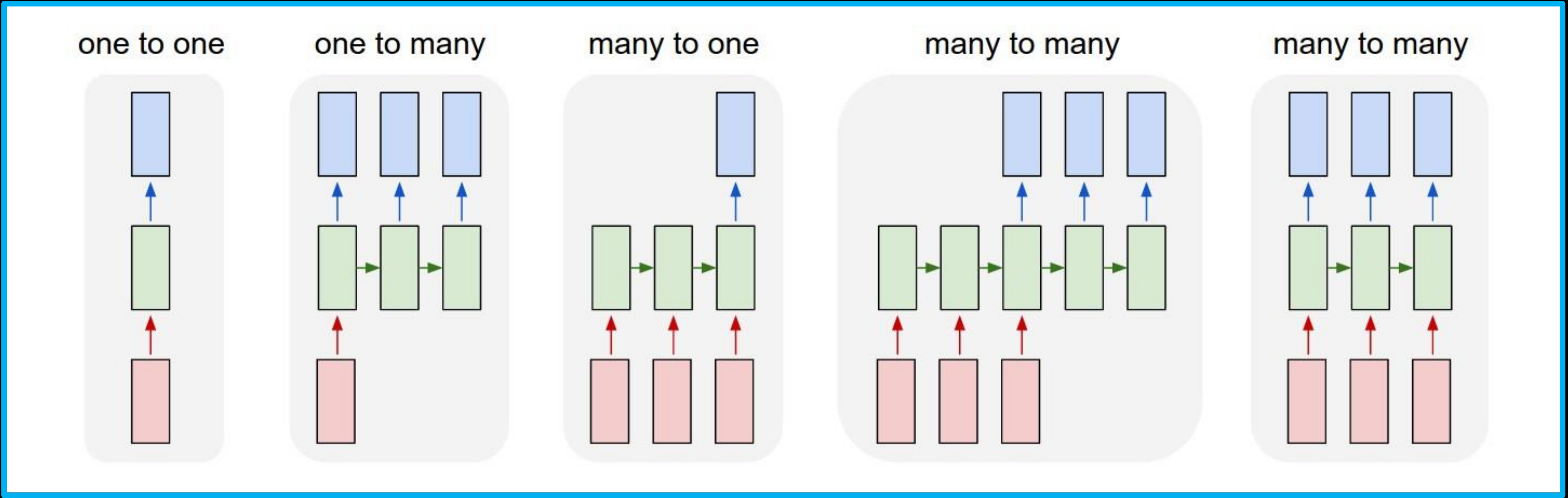


(3) Generation,
is a special example of many-to-many.

*) labels (y_i) can be easily generated – for each data sample the label will be the next item in the sequence

< Again, loss can include all of these distances

Types of models and data schemes



- Type of the model you would use depends on your data and the task you want to solve …

Illustration: [blog]

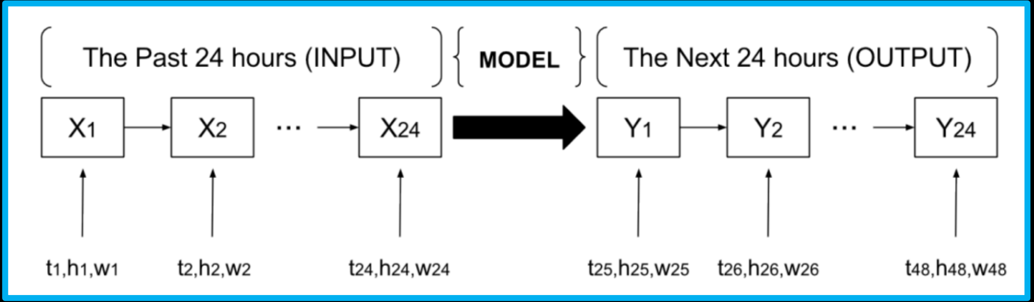
4 Examples from real-world research

- Weather forecasting

  Data:

  X = measured values (t_i, h_i, w_i)

  Y = known future values (t_{i+24}, h_{i+24}, w_{i+24})



t_i, h_i, w_i = temperature, humidity and wind speed

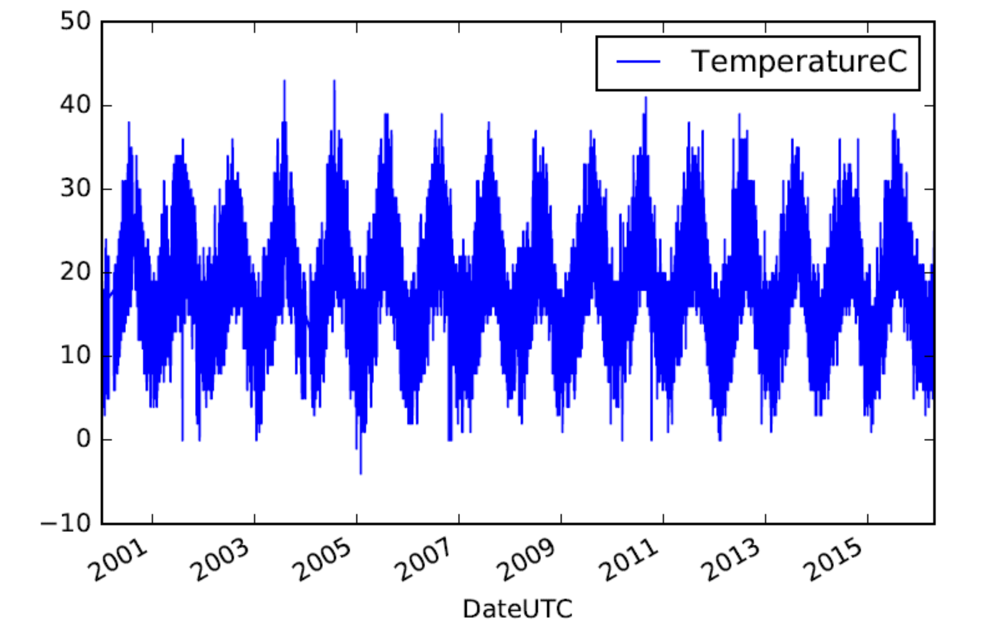


Fig. 5. 15 Years of Temperature values in Morocco, Tangier.

"Sequence to Sequence Weather Forecasting with Long Short-Term Memory RNNs" - Zaytar, M. A., et al. (2016)

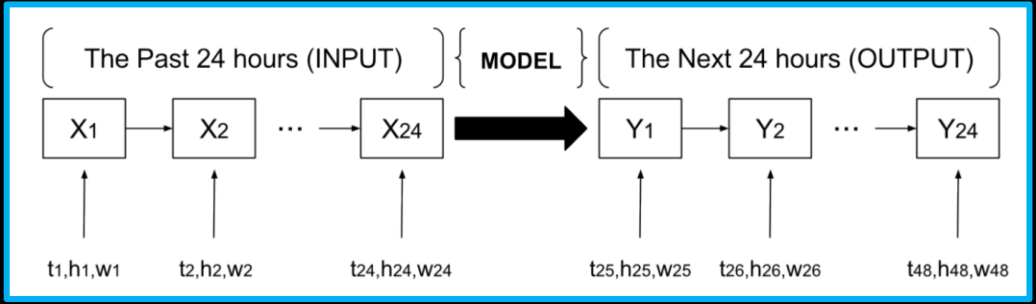
4 Examples from real-world research

- Weather forecasting

  Data:

  X = measured values (t_i, h_i, w_i)

  Y = known future values (t_{i+24}, h_{i+24}, w_{i+24})



t_i, h_i, w_i = temperature, humidity and wind speed

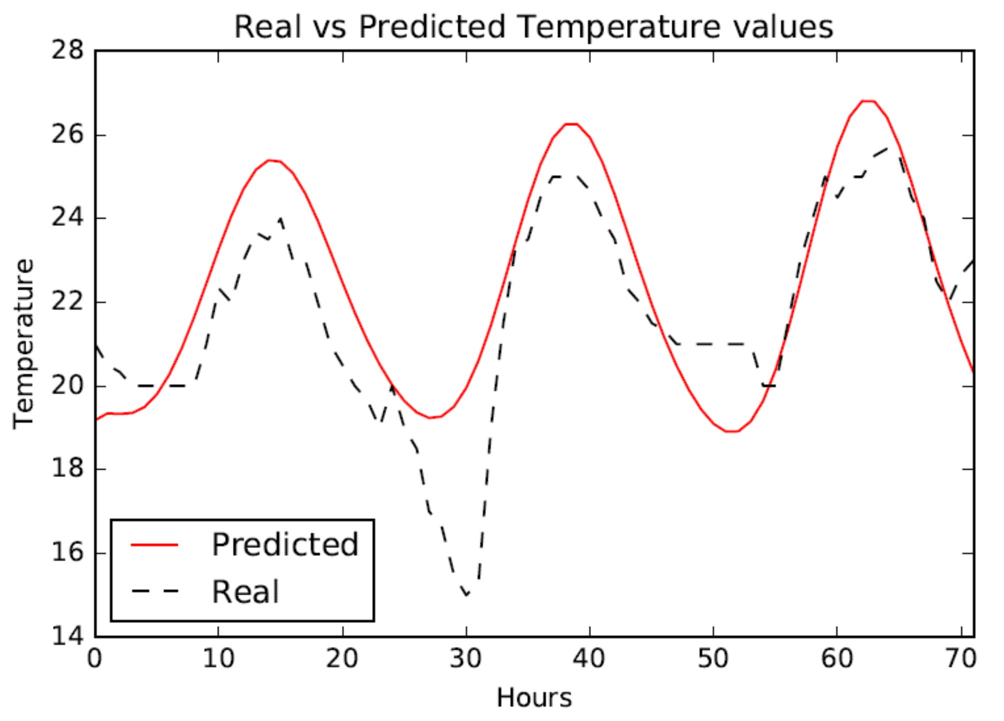


Fig. 10. Comparison of Temperature values for 72 hours.

"Sequence to Sequence Weather Forecasting with Long Short-Term Memory RNNs" - Zaytar, M. A., et al. (2016)

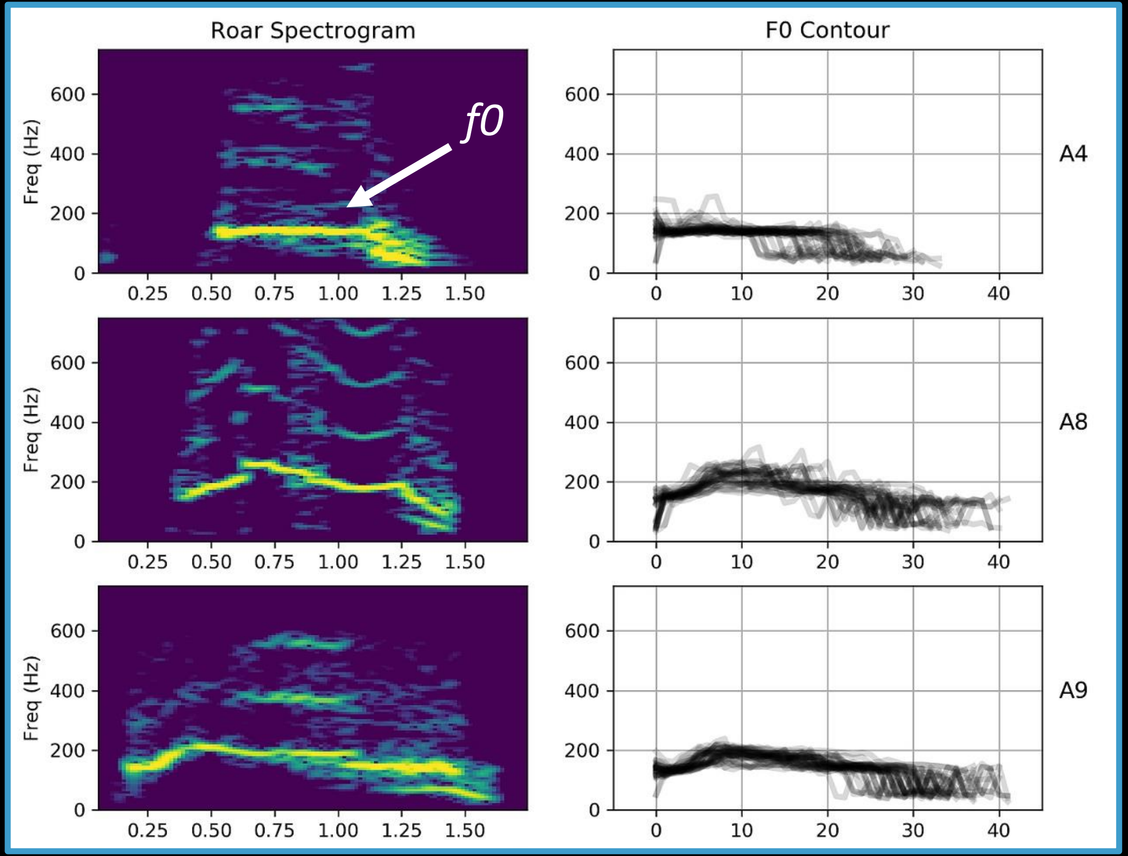
4 Examples from real-world research

- Lion roar identity classification (audio class.):

  Data:

  X = values from roar (f0 only), sequence of values over time

  Y = label for the lion (lion 1, 2, 3, …)



“Scientists discover the unique signature of a lion’s roar using machine learning” [Oxford 2020, link]

4 Examples from real-world research

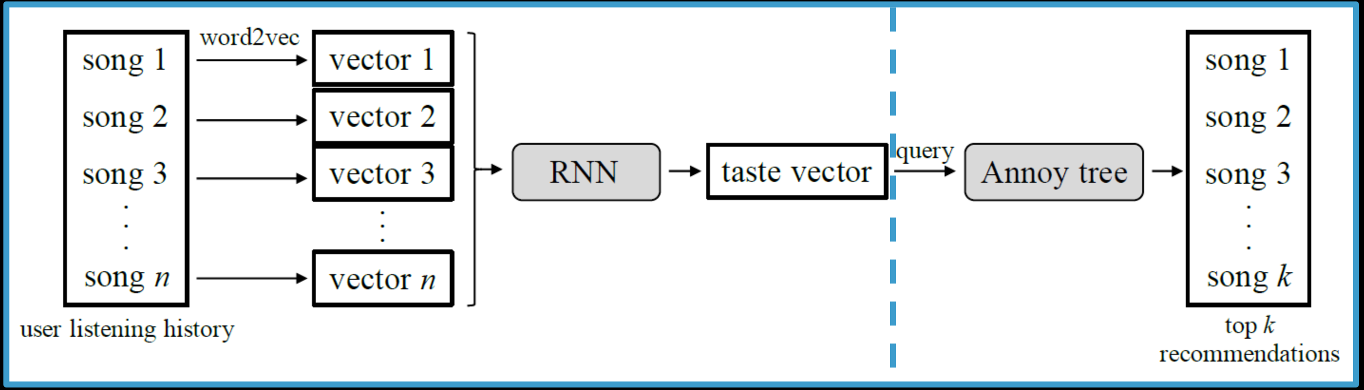
- Recommendation systems (Spotify)

  Data:

  X = listening history (tokenized)

  Y = *taste vectors* (from records of listened music)

  (Later uses "Annoy" ~ similar to nearest neighbours clustering methods – without labels)



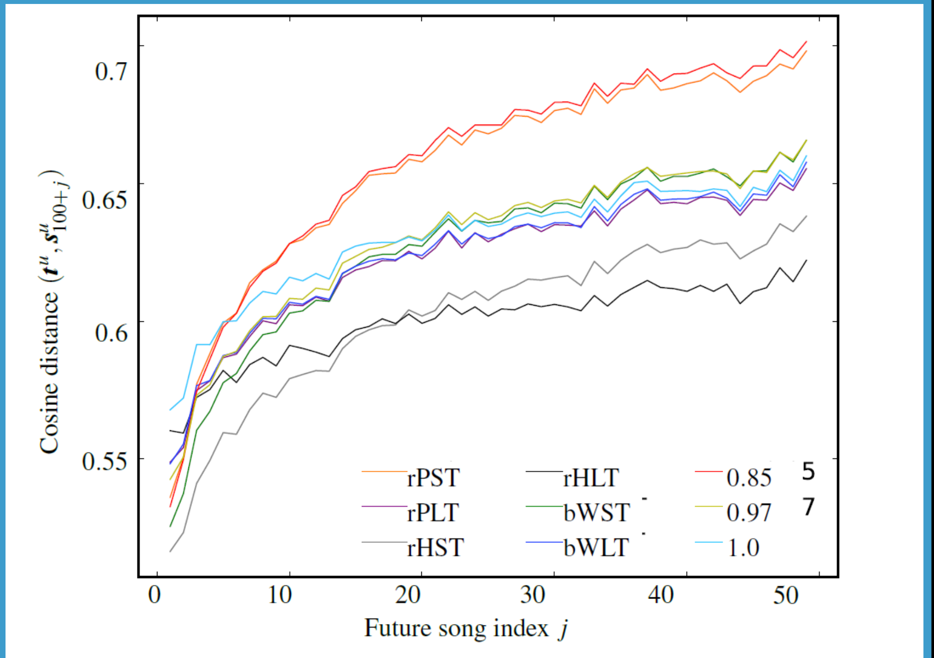


Fig. 5 Forward analysis of the taste vector models on filtered listening history data.

"Large-scale user modeling with RNNs for music discovery on multiple time scales." De Boom, Cedric, et al. (2018)

4 Examples from real-world research

- Liberal vs Illiberal rhetoric classification
  - X = speech notes, pre-processed, tokenized, etc.
  - Y = label derived from dictionary analysis (bellow)

- illiberalism vs liberalism
  - illiberalism
    - nationalism, paternalism
      - allah
      - almighty
      - anarch*
      - chaos
      - christ
      - christianity
      - christians
      - church
      - danger*
      - destabili*
      - evil
      - father*
      - god
      - hero*
  - liberalism
    - liberal values
      - authoritarian*
      - autocra*
      - choice
      - corrupt*
      - cruel*
      - demilitarization
      - dictator*
      - disarmament
      - discriminat*
      - diverse
      - diversity
      - equal*
      - fair*

Classification details, used keywords

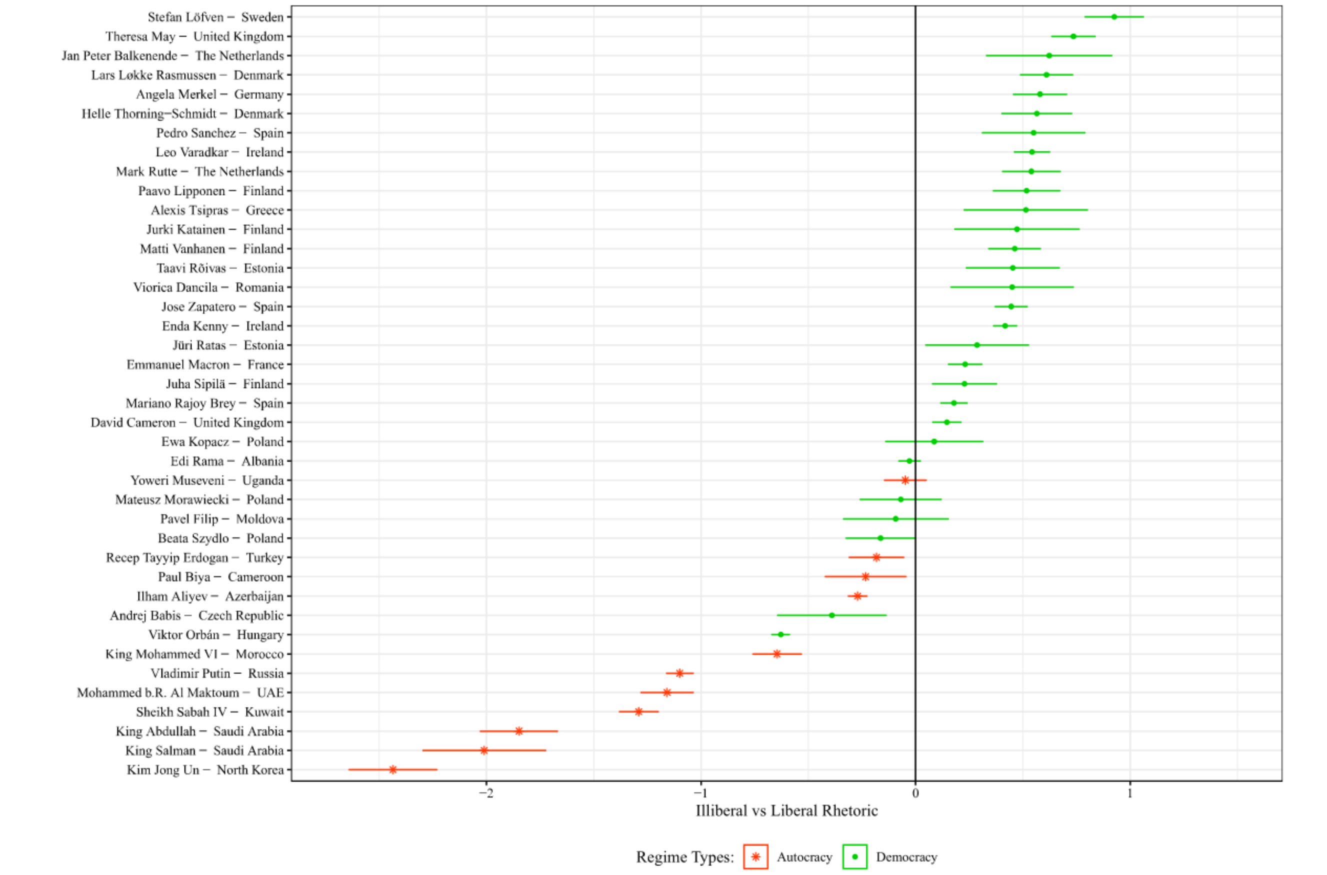


Fig. 1 Scaling the ideological content of all speeches per speaker based on the illiberal/liberal dictionary

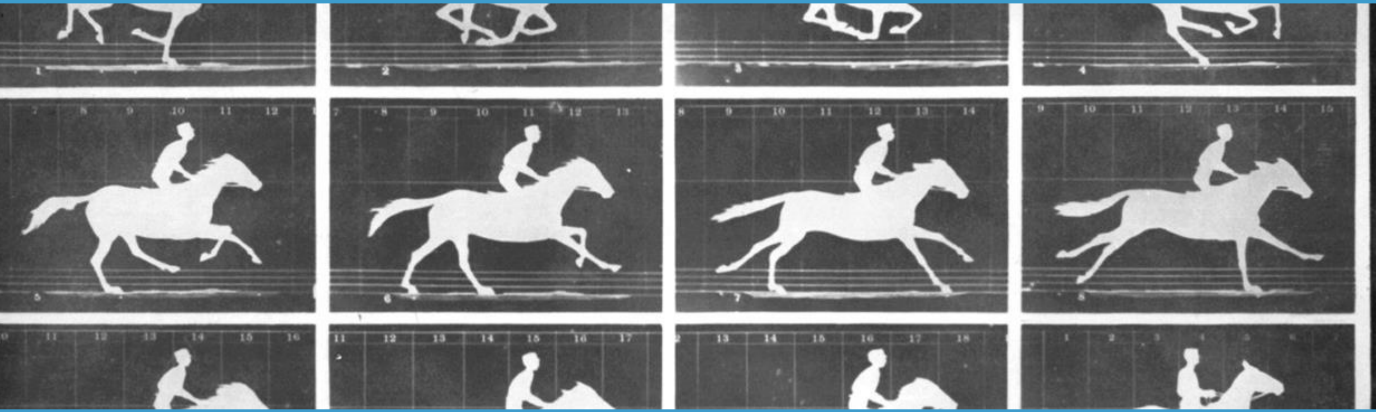
"Comparing public communication in democracies and autocracies: auto. text anal. of speeches by heads of government." ('19)

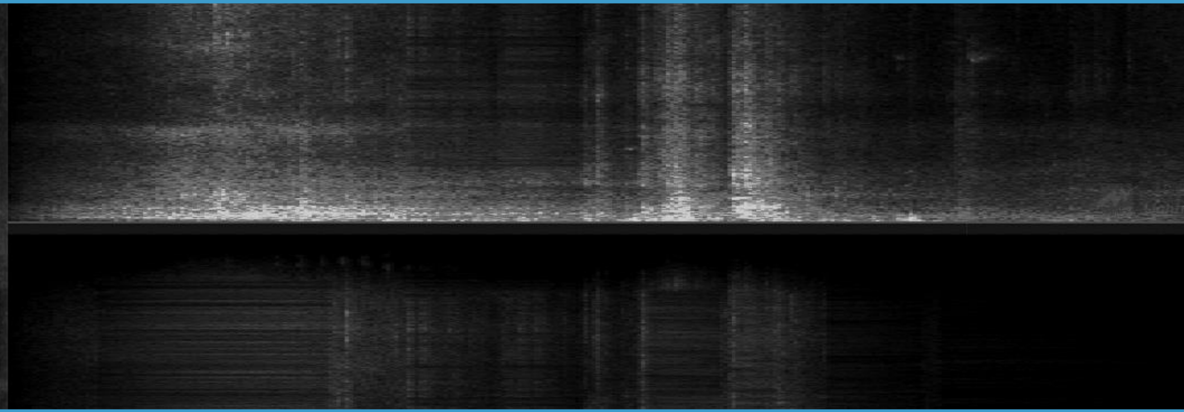
End of the lecture

***)** PS: follows material for the practical session …

AI for the Media

Week 5, Classifying Sequences





Practical: tweet sentiment analysis

~ Vít Růžička

Practical: Classifying Sequences

Tweet sentiment analysis

Interacting with the trained model

```
In [103]: custom_text = "The universe is a good place"
          #custom_text = "The universe is a bad place"

          x_custom = pad_sequences(tokenizer.texts_to_sequences([custom_text]),
                                   maxlen = MAX_SEQUENCE_LENGTH)

          print("^ this gets represented as:", x_custom)

^ this gets represented as: [[    0    0    0    0    0    0    0    0    0
     0    0    0    0    0    0    0    0    0    0    0    0    0 2698
     1  305]]
```

```
In [104]: scores_custom = model_ours.predict(x_custom, verbose=1, batch_size=10000)
          print("model prediction:", scores_custom)
          y_pred_1d_custom = [decode_sentiment(score) for score in scores_custom]
          print("decoded:", y_pred_1d_custom)

1/1 [==============================] - 0s 224ms/step
model prediction: [[0.8245092]]
decoded: ['Positive']
```

Continue with code on Github:

- Repo: github.com/previtus/cci_AI_for_the_Media
- **Notebook** directly: aim05_twitter-sentiment-analysis.ipynb

- *I will put my lectures and code there (it's going to be easier to use the Colab demos from a public repo)*

The end