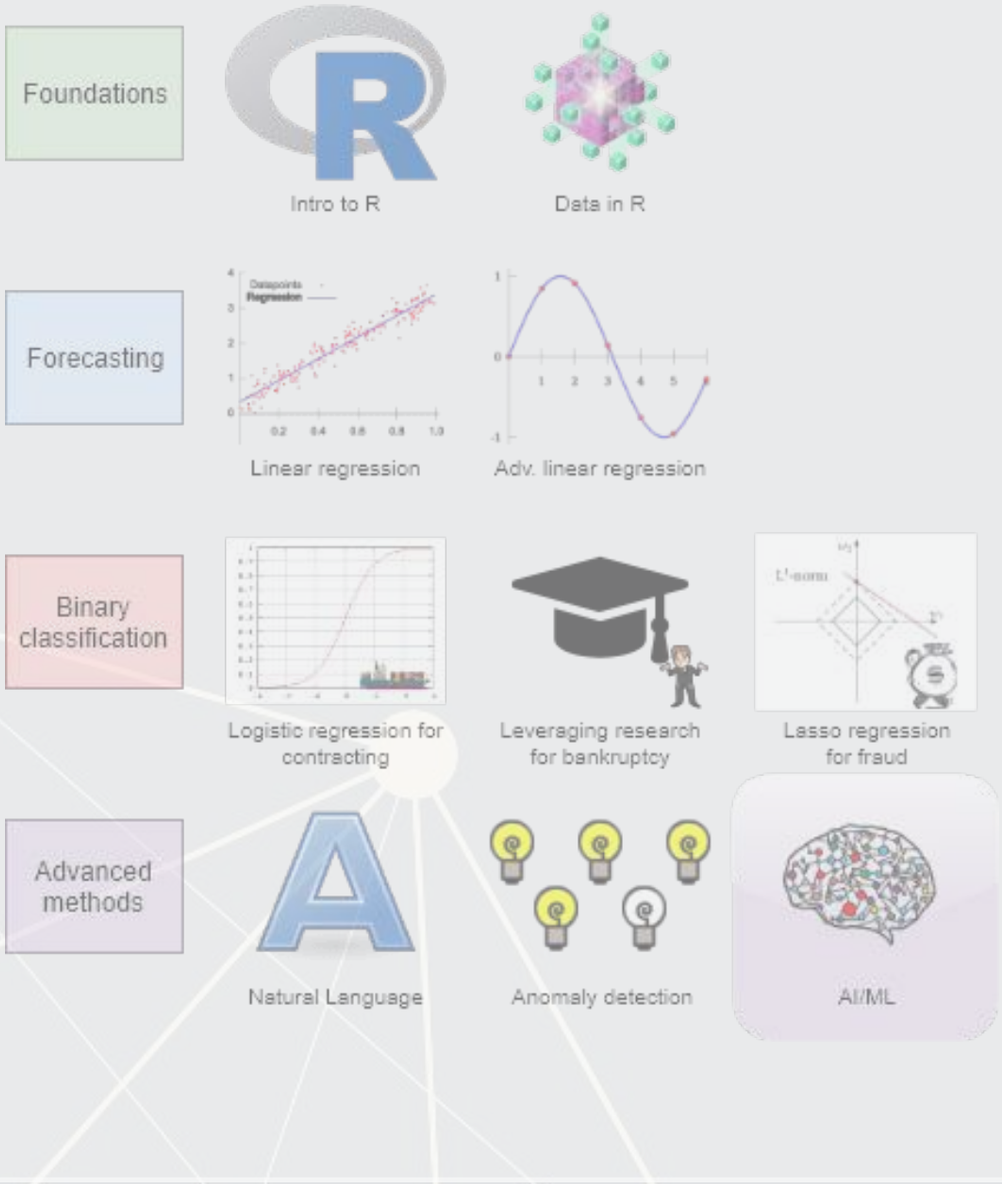
ACCT 420: Machine Learning and AI

Session 11

Dr. Richard M. Crowley

Front matter

Learning objectives



- **Theory:**
  - Neural Networks
- **Application:**
  - Varied
- **Methodology:**
  - Vector methods
  - 6 types of neural networks
  - Others

Group project

- Almost done!
  - Last submission deadline is tomorrow night
- On Tuesday, you will have an opportunity to present your work
  - 12-15 minutes
- You will also need to submit your report & code on Tuesday
  - Please submit as a zip file
  - Be sure to include your report AND code
    - Code should cover your final model
      - Covering more is fine though

Final homework

- Strong demand for a later due date, so I'll push it back to November 20th (11:59pm)
  - Note: To cover this, I will release a set of slides that:
    - Summarizes the homework
    - Addresses the most common mistakes
  - Take a look at the slides when they are posted!

> Due by the end of November 20th

Final exam

- Still preparing
  - Format will be as stated:
    - ~30% Multiple choice related to coding
    - ~70% Long format
- For studying
  - I will provide a solved case on Enron, which can serve as a study guide of sorts for the forensics part of the class
  - I will try to provide some sample questions *after the final is written*
    - This way I can
- The best way to study is to practice
  - Your group projects are an example of this
  - Consider working out another problem on your own or with a group, of your choice
    - Is there anything you ever wanted to know about businesses?
    - Feel free to schedule a consultation to go over your findings

Languages for ML/AI

R for ML/AI

Older methods

- `caret`
- `randomForest`
- `nnet`
- `e1071`

Best-in-class

- `glmnet`: LASSO and elastic nets
- `xgboost`: XGBoost
- `Prophet`: ML for time series forecasting
- `keras`: Plugs into python's Keras
- `H2O4GPU`: Plugs into python's H2O
- `spacyr`: Plugs into python's SpaCy

Python for ML/AI

Older methods

- Sci-kit learn – one stop shop for most older libraries
- RPy2
- scipy + numpy + pandas + statsmodels
  - Add Theano in for GPU compute

Best-in-class

- **TENSORFLOW** (Google)
  - Can do everything
- pytorch – python specific Torch port
- gensim: “Topic modelling for humans”
- H2O (H2O)
- caffe (Berkley)
- caffe2 (Facebook)
- SpaCy – Fast NLP processing
- CoreNLP – through various wrappers to the Java library

Others for ML/AI

- C/C++: Also a first class language for TensorFlow!
  - Really fast – precompiled
  - Much more difficult to code in
- Swift: Strong TensorFlow support
- Javascript: Improving support from TensorFlow and others

Why do I keep mentioning TensorFlow?

- It can run almost ANY ML/AI/NN algorithm
- It has APIs for easier access like Keras
- Comparatively easy GPU setup
- It can deploy anywhere
  - Python & C/C++ built in
  - Swift and R Bindings for Haskell, R, Rust, Swift
  - TensorFlow light for mobile deployment
  - TensorFlow.js for web deployment

TensorFlow Lite

TensorFlow.js

magenta

TensorFlow Hub

Why do I keep mentioning TensorFlow?

- It has strong support from Google and others
  - TensorFlow Hub – Premade algorithms for text, image, and video
  - tensorflow/models – Premade code examples
    - The research folder contains an amazing set of resources
  - tensorflow/tensor2tensor – AI research models

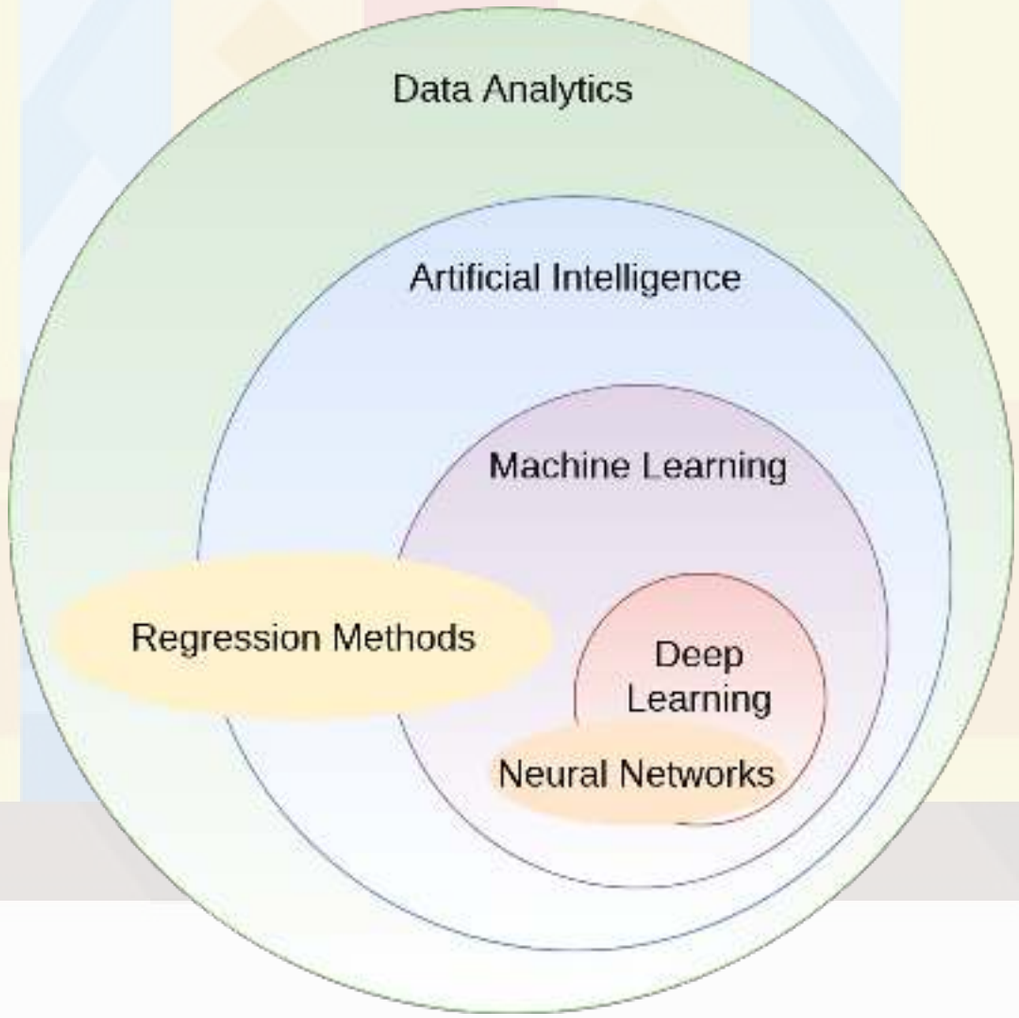
Other notable frameworks

- Caffe
  - Python, C/C++, Matlab
  - Good for image processing
- Caffe2
  - C++ and Python
  - Still largely image oriented
- Microsoft Cognitive Toolkit
  - Python, C++
  - Scales well, good for NLP
- Torch and Pytorch
  - For Lua and python
  - fast.ai, ELF, and AllenNLP
- H20
  - Python based
  - Integration with R, Scala…

Neural Networks

What are neural networks?

- The phrase *neural network* is thrown around almost like a buzz word
- *Neural networks* are actually a specific type class algorithms
  - There are many implementations with different primary uses



What are neural networks?

- Originally, the goal was to construct an algorithm that behaves like a human brain
  - Thus the name
- Current methods don't quite reflect human brains, however:
  1. We don't fully understand how our brains work, which makes replication rather difficult
  2. Most neural networks are constructed for specialized tasks (not general tasks)
  3. Some (but not all) neural networks use tools our brain may not have
     - I.e., back propogation is potentially possible in brains, but it is not pinned down how such a function occurs (if it does occur)

What are neural networks?

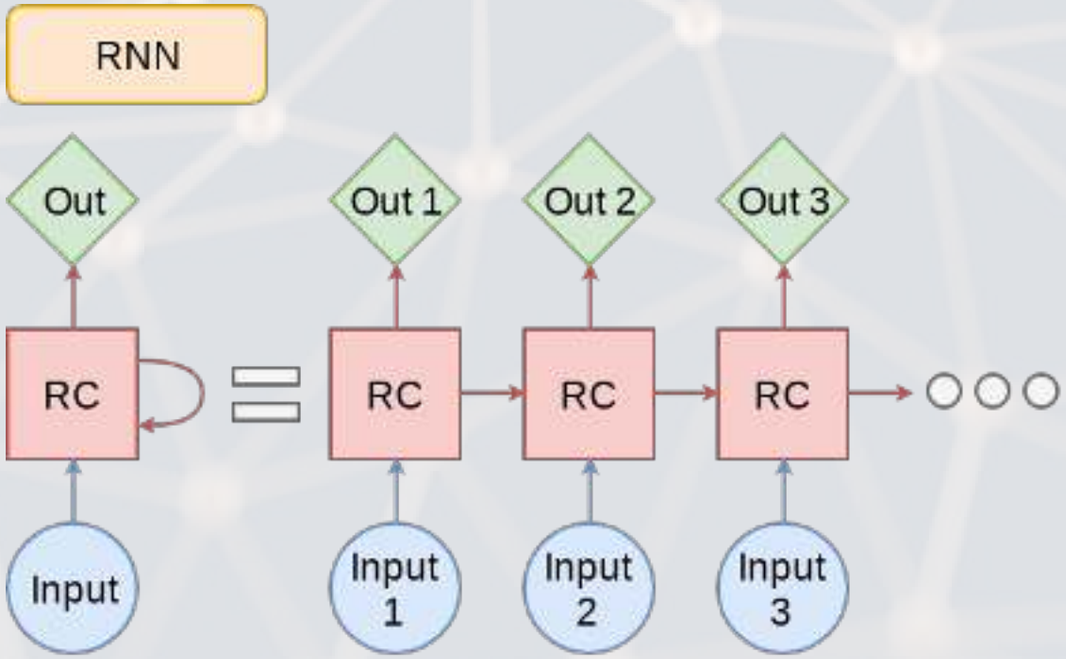
- Neural networks are a method by which a computer can learn from observational data
- In practice:
  - They were not computationally worthwhile until the mid 2000s
  - They have been known since the 1950s (perceptrons)
  - They can be used to construct algorithms that, at times, perform better than humans themselves
    - But these algorithms are often quite computationally intense, complex, and difficult to understand
  - Much work has been and is being done to make them more accessible

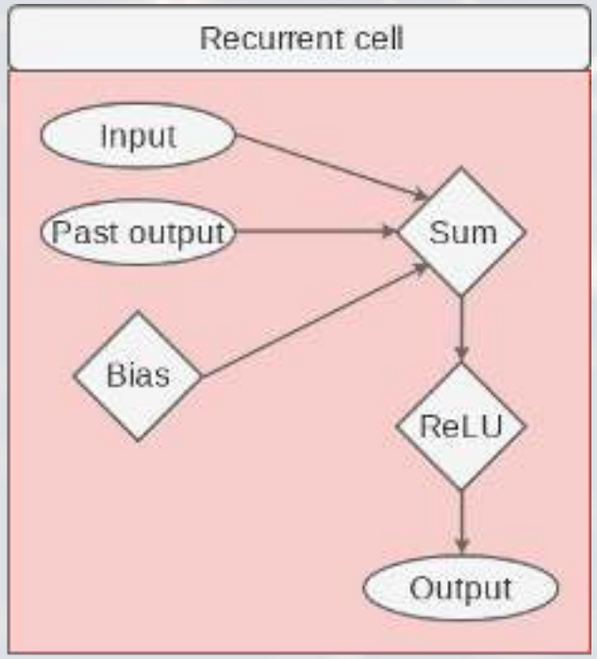
Types of neural networks

- There are *a lot* of neural network types
  - See The "Neural Network Zoo"
- Some of the more interesting ones which we will see or have seen:
  - RNN: Recurrent Neural Network
  - LSTM: Long/Short Term Memory
  - CNN: Convolutional Neural Network
  - DAN: Deep Averaging Network
  - GAN: Generative Adversarial Network
- Others worth noting
  - VAE (Variational Autoencoder): Generating *new* data from datasets

RNN: Recurrent NN

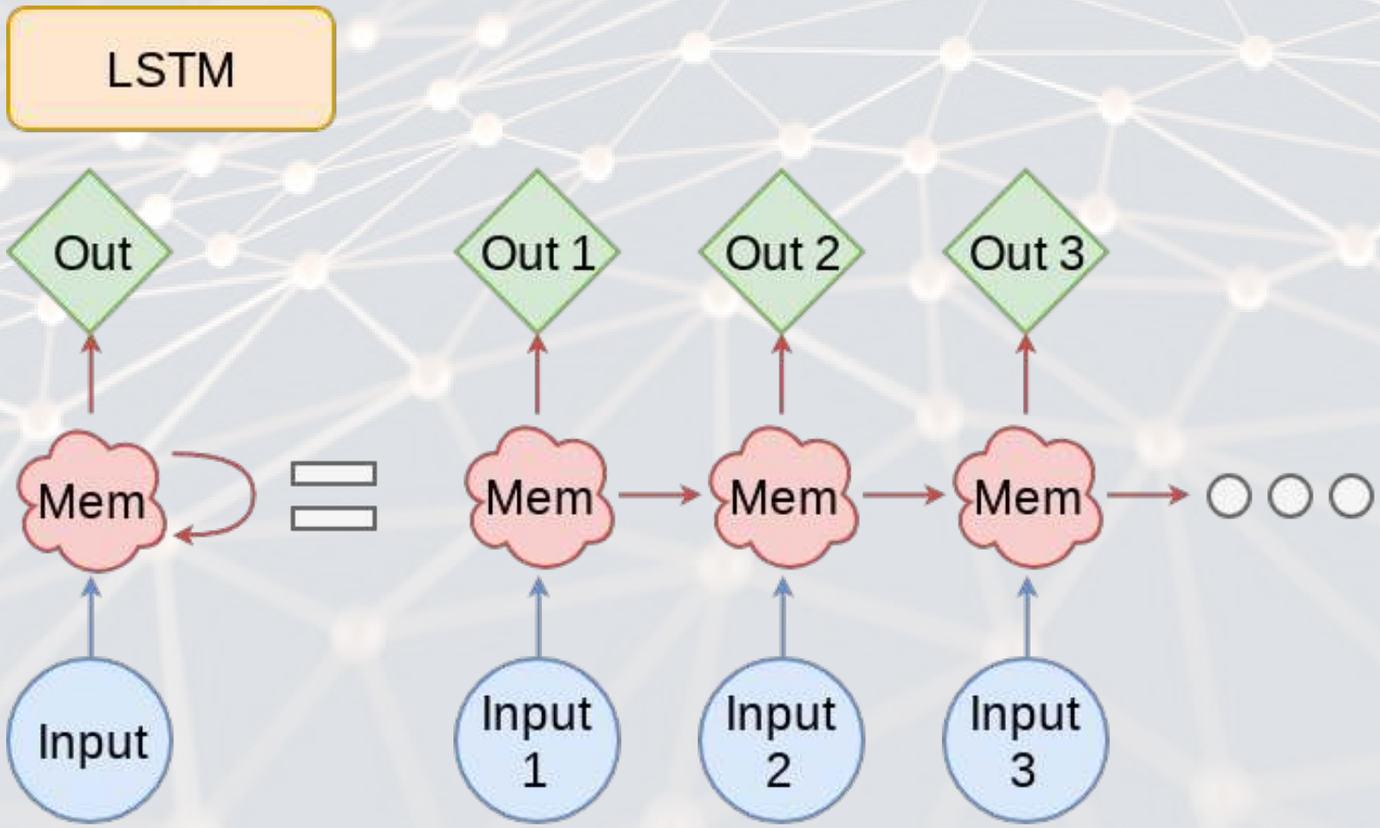
- Recurrent neural networks embed a history of information in the network
  - The previous computation affects the next one
  - Leads to a *short term memory*
- Used for speech recognition, image captioning, anomaly detection, and many others
  - Also the foundation of LSTM
  - SketchRNN

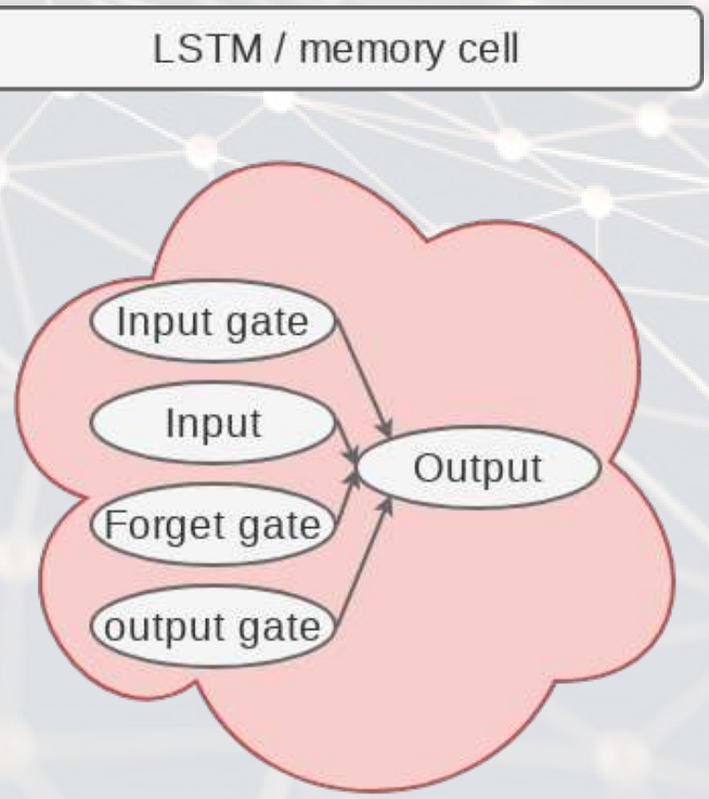




LSTM: Long Short Term Memory

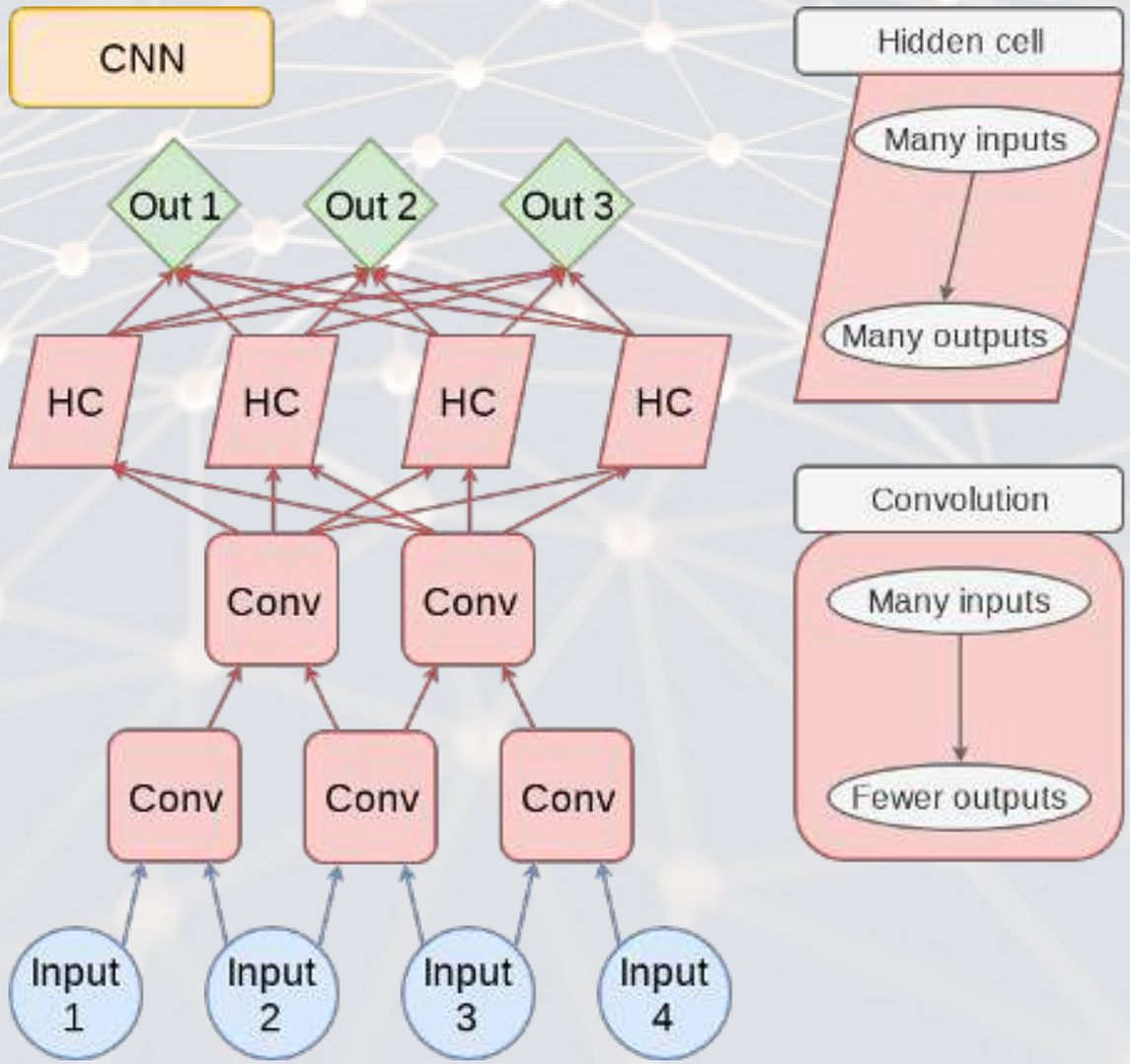
- LSTM improves the *long term memory* of the network while explicitly modeling a *short term memory*
- Used wherever RNNs are used, and then some
  - Ex.: Seq2seq (machine translation)





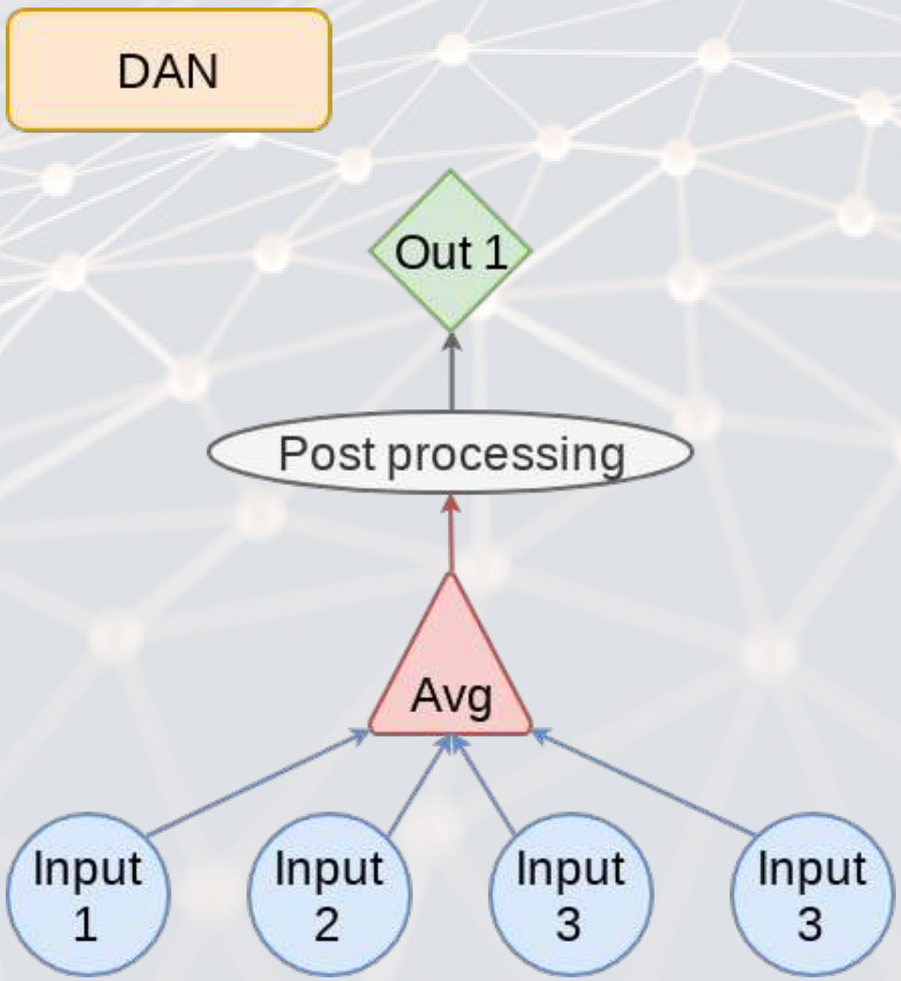
CNN: Convolutional NN

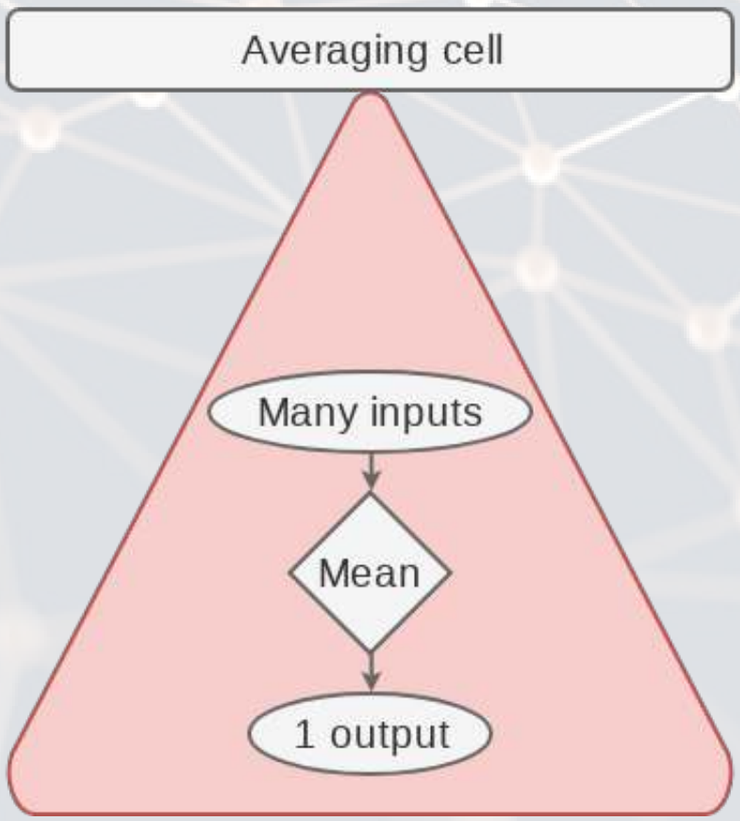
- Networks that excel at object detection (in images)
- Can be applied to other data as well
- Ex.: Inception



DAN: Deep Averaging Network

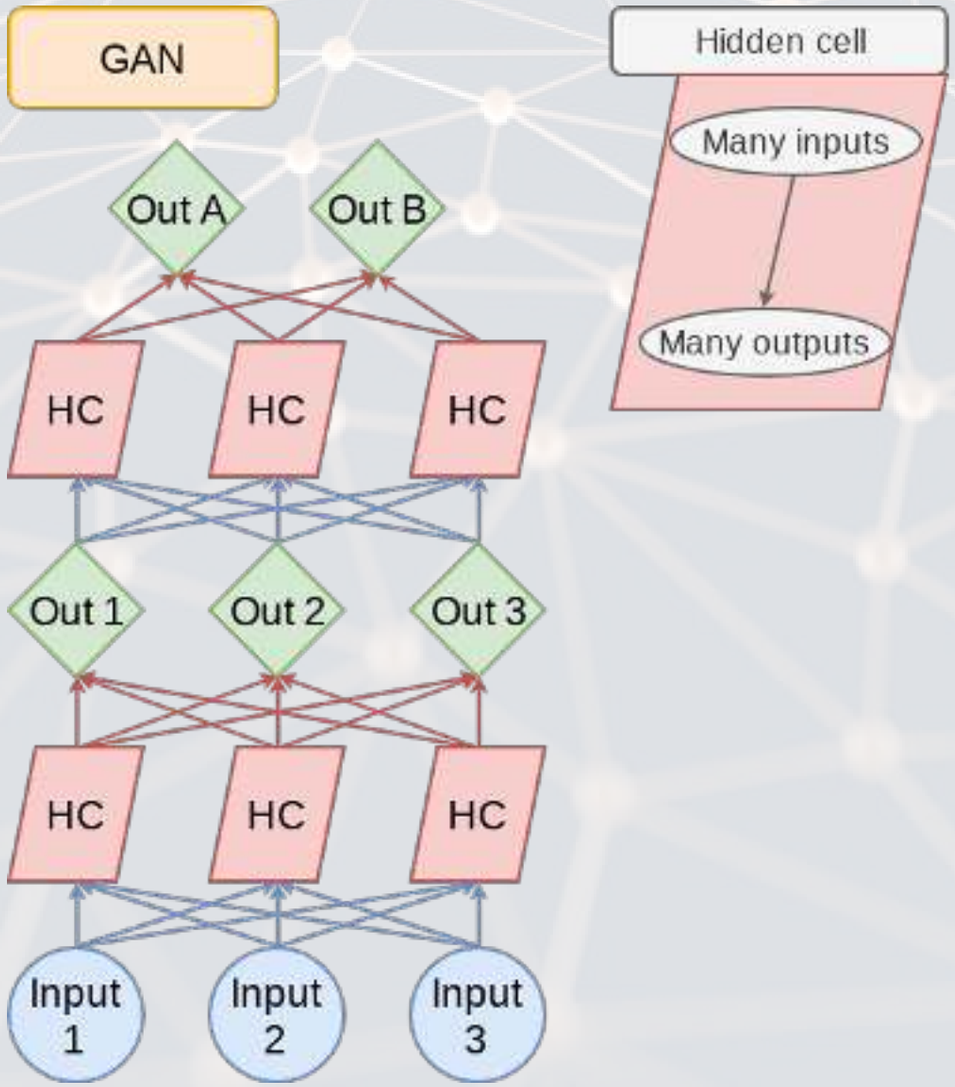
- DANs are simple networks that simply average their inputs
- Averaged inputs are then processed a few times
- These networks have found a home in NLP
  - Ex.: Universal Sentence Encoder





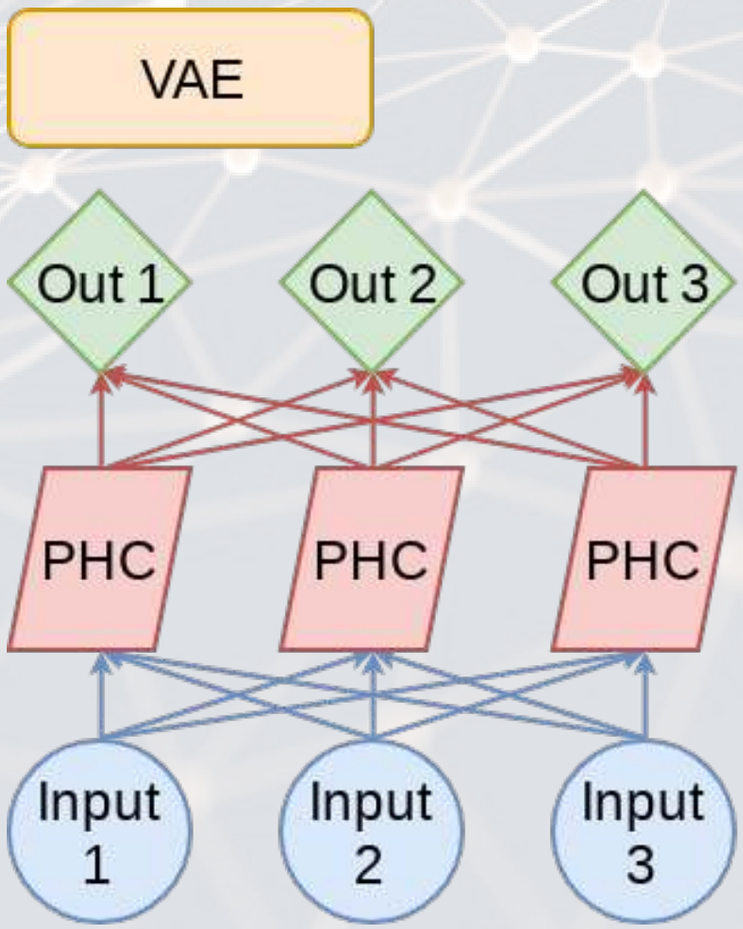
GAN: Generative Adversarial Network

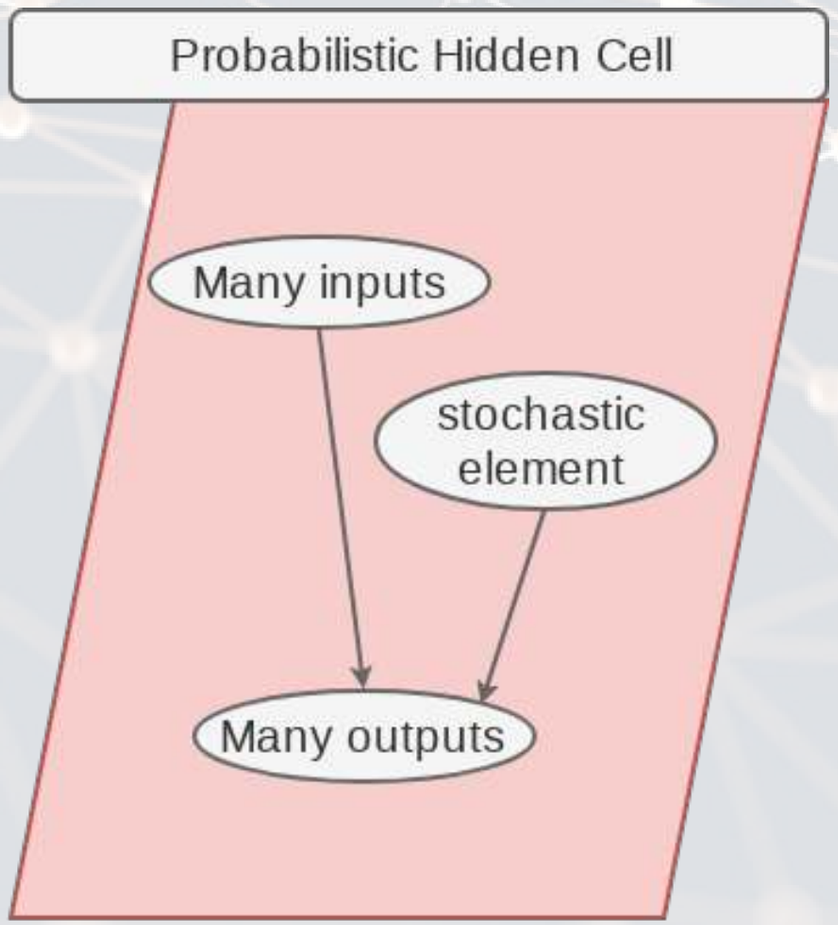
- Feature two networks working against each other
- Many novel uses
  - Ex.: The anonymization GAN from last week
  - Ex.: Aging images



VAE: Variational Autoencoder

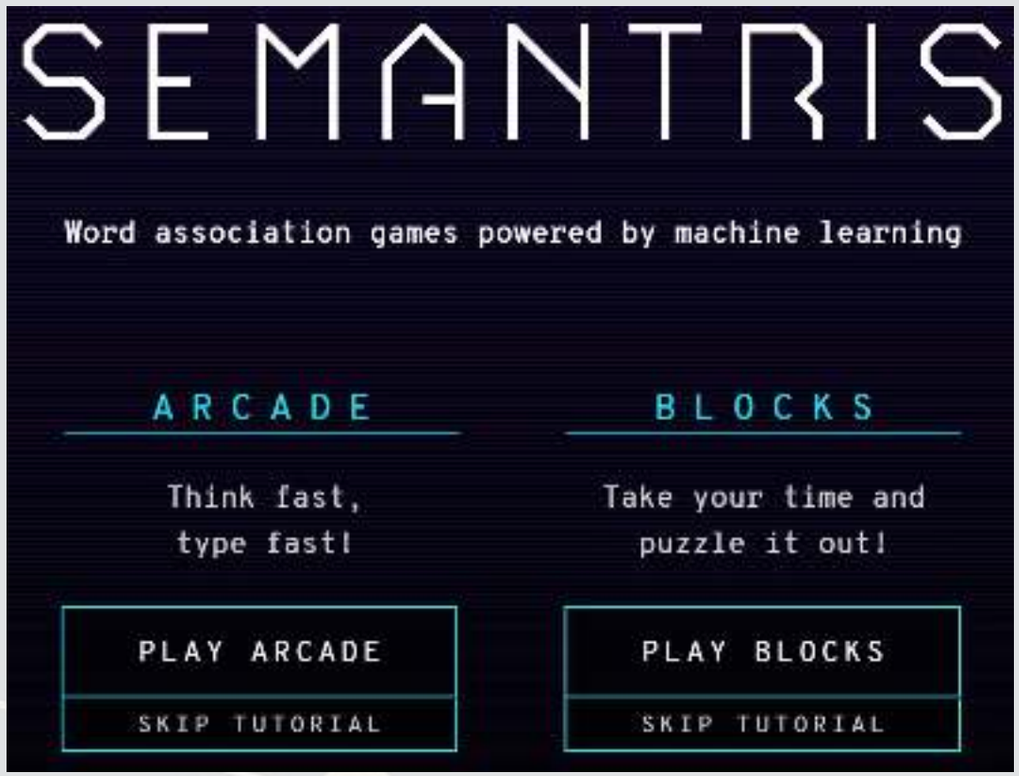
- An autoencoder (AE) is an algorithm that can recreate input data
- Variational means this type of AE can vary other aspects to generate completely new output
  - Good for creating fake data
- Like a simpler, noisier GAN

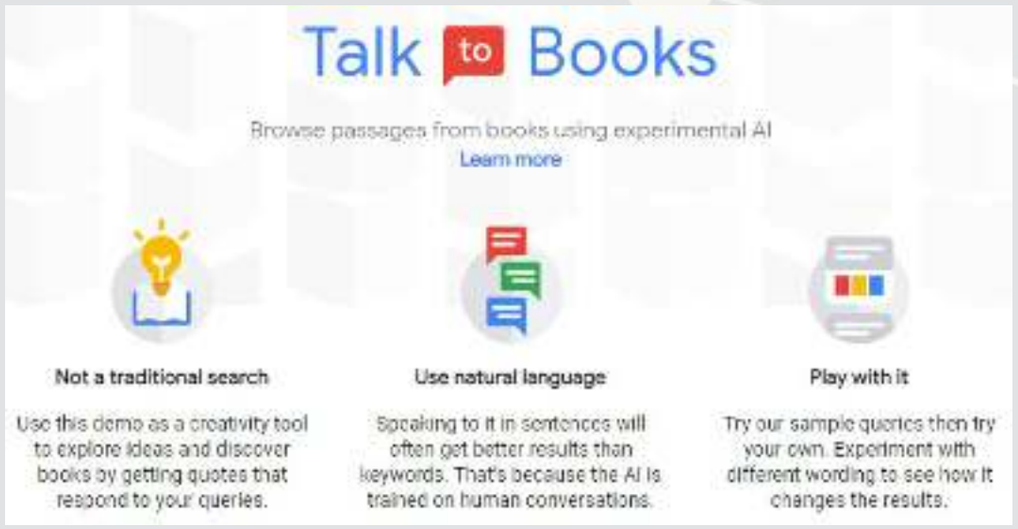




Vector space models

Motivating examples





What are “vector space models”

- Different ways of converting some abstract information into numeric information
  - Focus on maintaining some of the underlying structure of the abstract information
- Examples (in chronological order):
  - Word vectors:
    - Word2vec
    - GloVe
  - Paragraph/document vectors:
    - Doc2Vec
  - Sentence vectors:
    - Universal Sentence Encoder

Word vectors

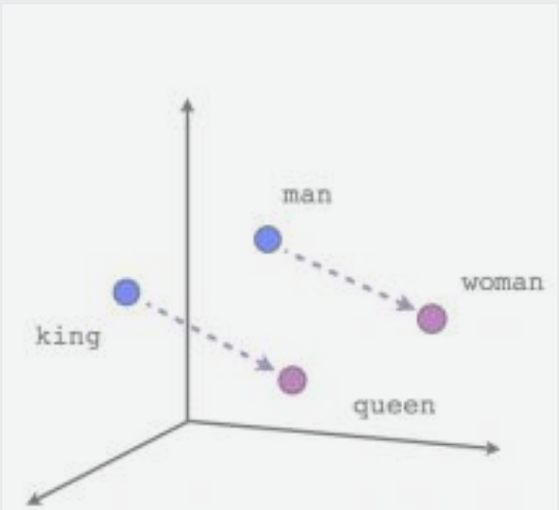
- Instead of coding individual words, encode word meaning
- The idea:
  - Our old way (encode words as IDs from 1 to N) doesn't understand relationships such as:
    - Spatial
    - Categorical
    - Grammatical (weakly when using stemming)
    - Social
    - etc.
  - Word vectors try to encapsulate all of the above
    - They do this by encoding words as a vector of different features

Word vectors: Simple example

words	f_animal	f_people	f_location
dog	0.5	0.3	-0.3
cat	0.5	0.1	-0.3
Bill	0.1	0.9	-0.4
turkey	0.5	-0.2	-0.3
Turkey	-0.5	0.1	0.7
Singapore	-0.5	0.1	0.8

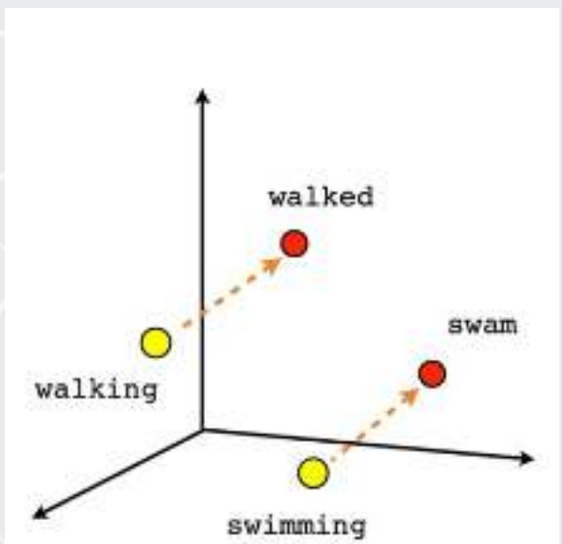
- The above is an idealized example
- Notice how we can tell apart different animals based on their relationship with people
- Notice how we can distinguish turkey (the animal) from Turkey (the country) as well

What it retains: word2vec



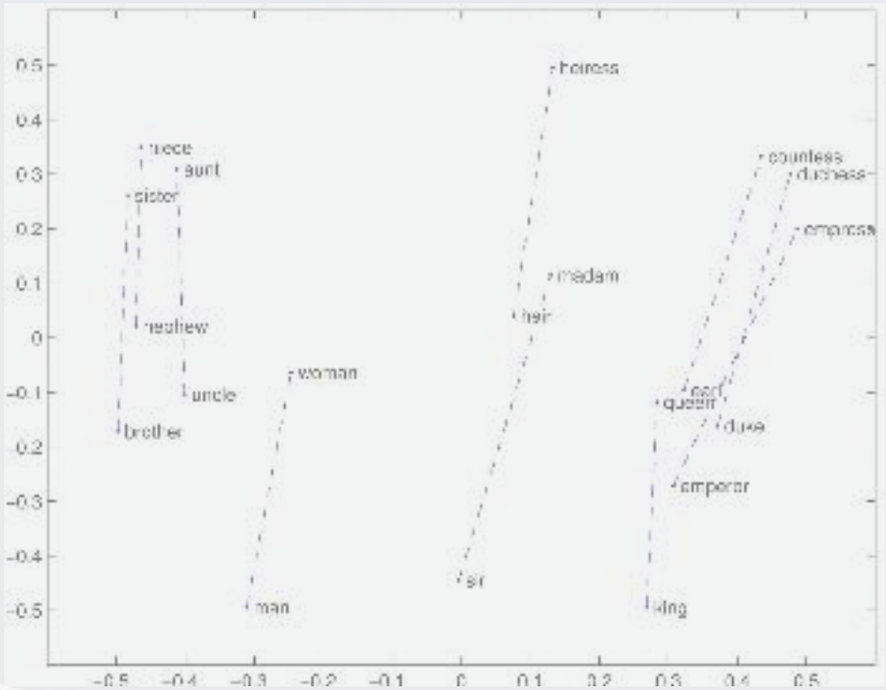
Male-Female

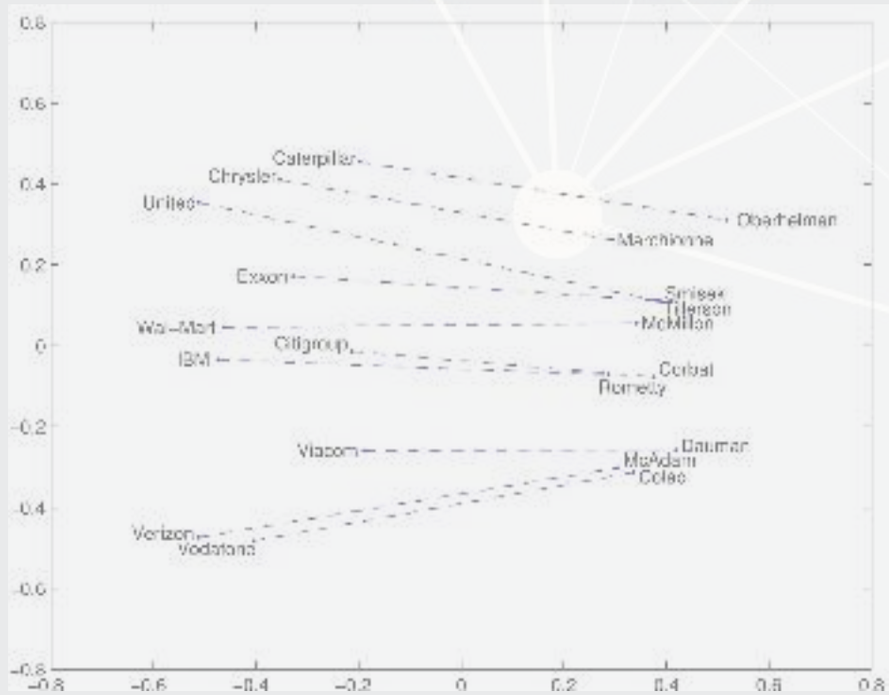
Country-Capital

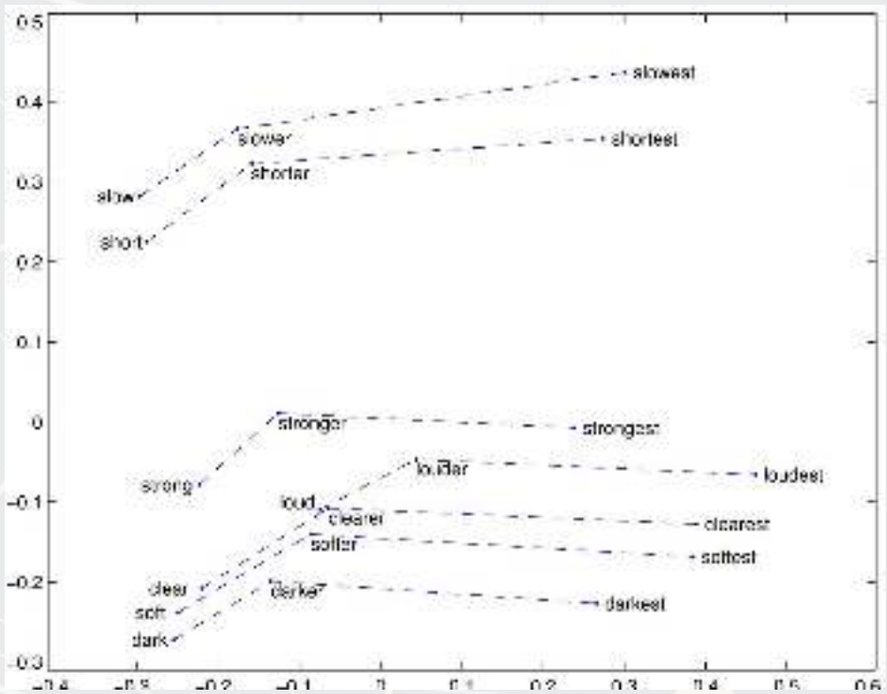


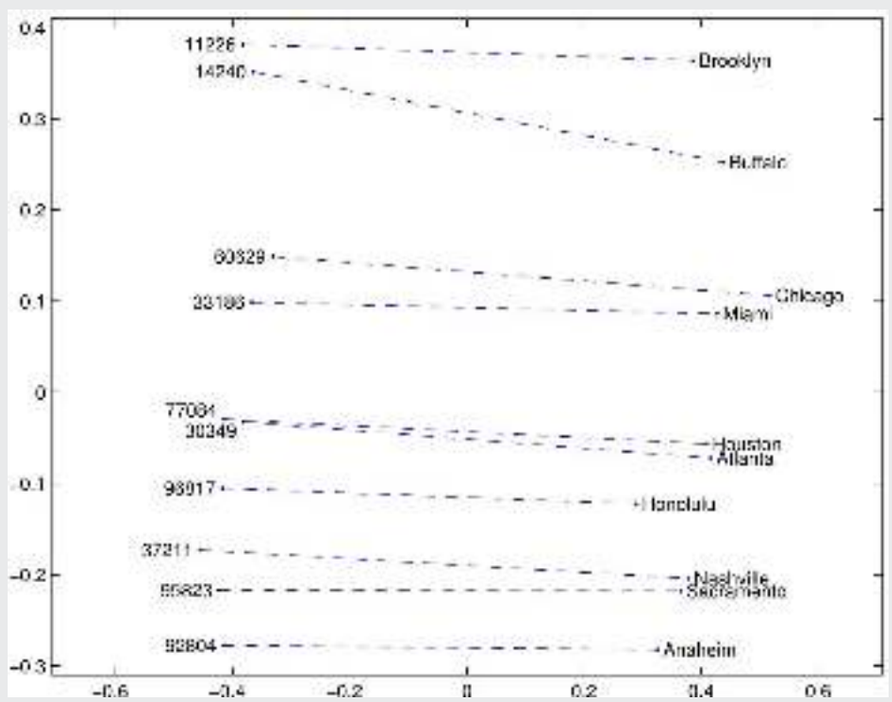
Verb tense

What it retains: GloVe







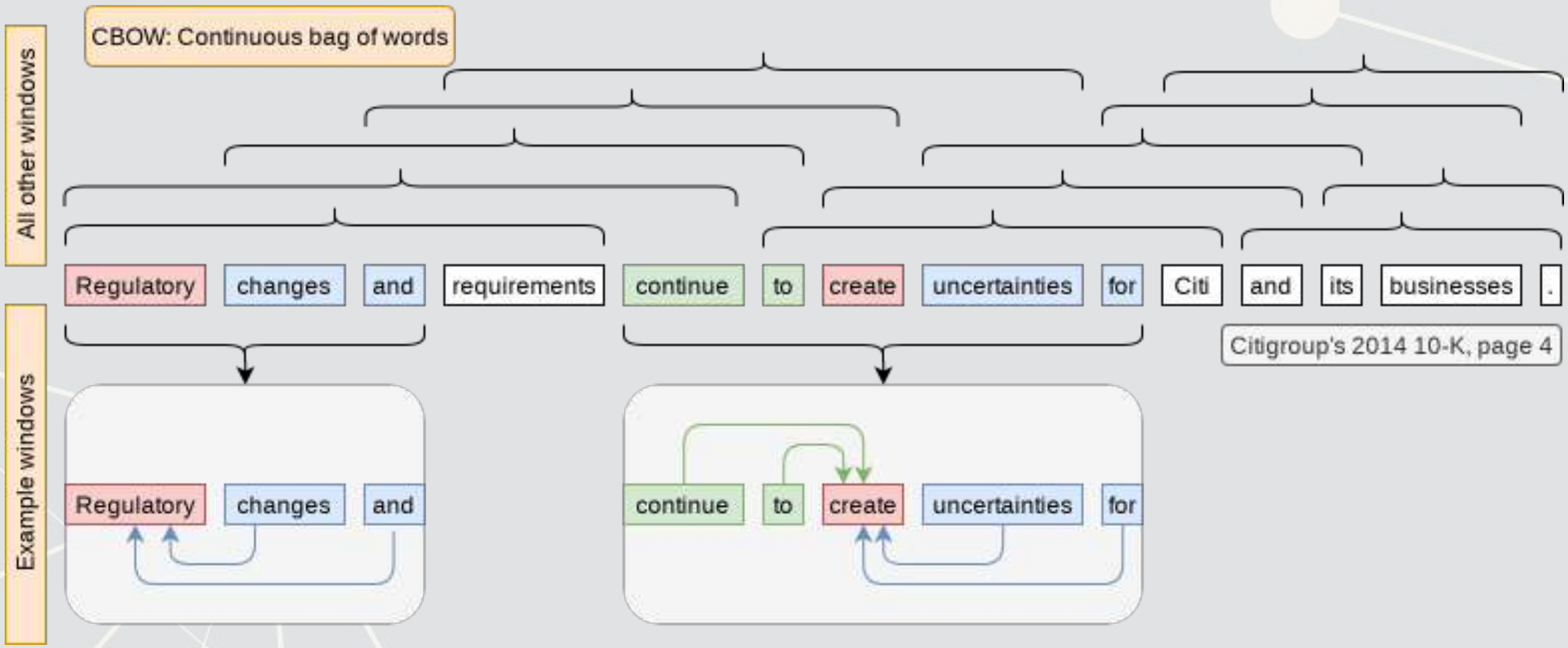


How to build word vectors

- Two ways:
  1. Word co-occurrence (like how LDA worked)
     - Global Vectors (GloVe) works this way
       - Available from the `text2vec` package
  2. Word order (using an NN)
     - word2vec works this way
       - Available from the `rword2vec` package
       - Uses a 2 layer neural network

How does word order work?

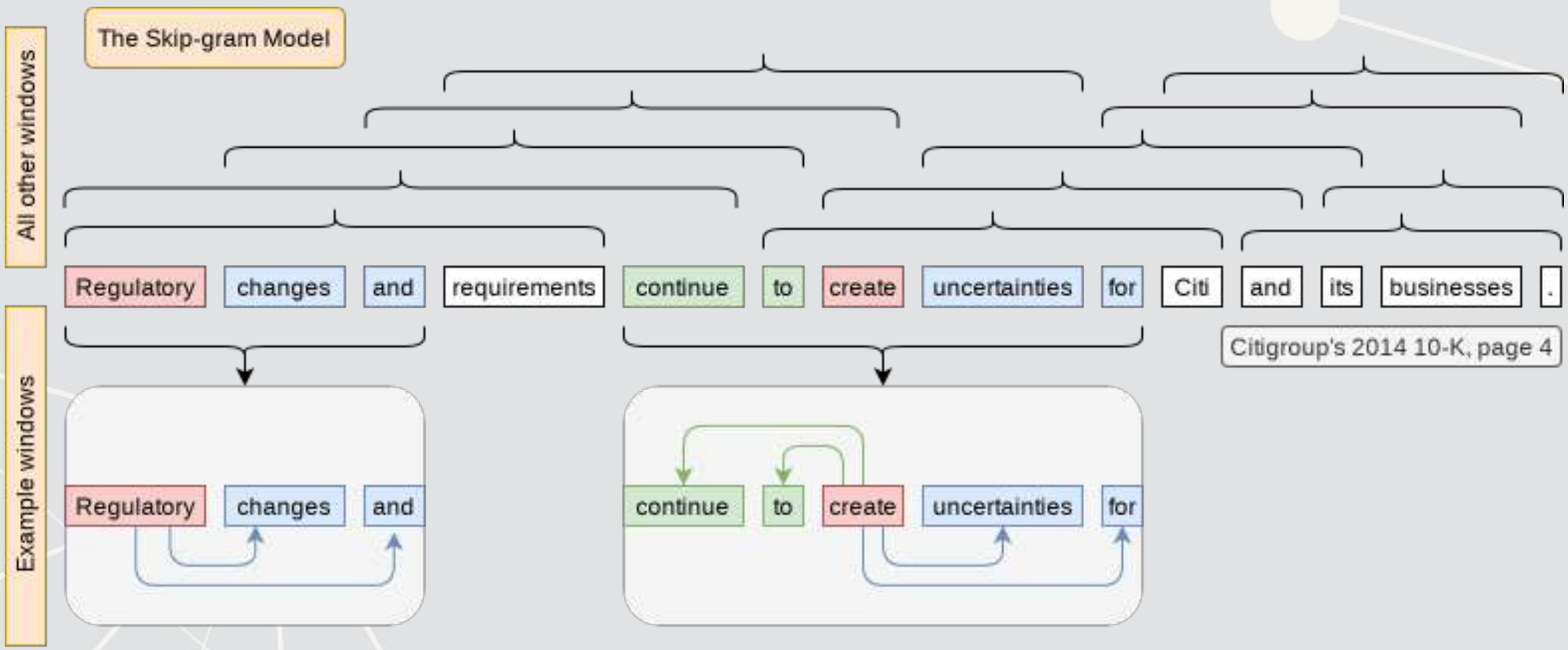
Infer a word's meaning from the words around it



Refered to as CBOW (continuous bag of words)

How else can word order work?

Infer a word's meaning by *generating* words around it



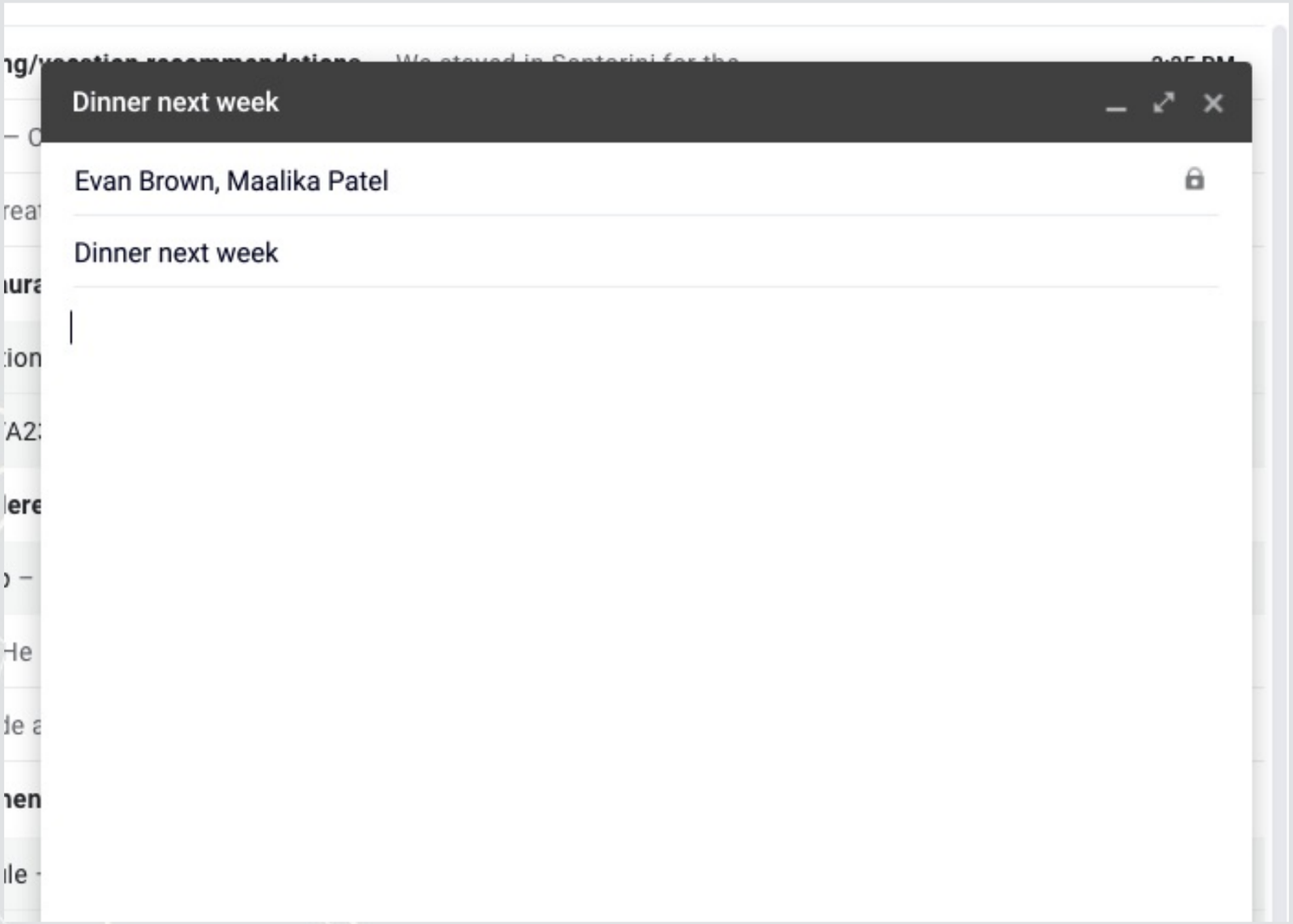
Refered to as the Skip-gram model

Document vectors

- Document vectors work very similarly to word vectors
  - 1 added twist: a document/paragraph/sentence level factor variable
    - This is used to learn a vector representation of each text chunk
    - Learned simultaneously with the word vectors
      - Caveat: it can also be learned independently using PV-DBOW
- This is quote related to what we learned with LDA as well!
  - Both can tell us the topics discussed

Universal Sentence Encoder (USE)

- We saw this briefly last week
  - This is the algorithm with less bias
- Focused on representing sentence-length chunks of text



A fun example of with USE

- Predict Shakespeare with Cloud TPUs and Keras

Hamlet		
Starting text	Original text	Machine generated (USE)
SCENE I. Elsinore. A platform before the Castle. Enter Francisco and Barnardo, two sentinels. BARNARDO. Who's there? FRANCISCO. Nay, answer me. Stand and unfold yourself. BARNARDO. Long live the King!	FRANCISCO. Barnardo? BARNARDO. He. FRANCISCO. You come most carefully upon your hour. BARNARDO. 'Tis now struck twelve. Get thee to bed, Francisco. FRANCISCO. For this relief much thanks. 'Tis bitter cold, And I am sick at heart. BARNARDO. Have you had quiet gua...	MARLAND. So count what? Make haste. THESEUS. Where her an unchaste Marcius may wonder such sad, When dues? LANCASTER. I thank you. Ler. HELEN. And honest Prince, faith, sweet and Exeter. LUCENTIO. It wrose rest your measure made at...

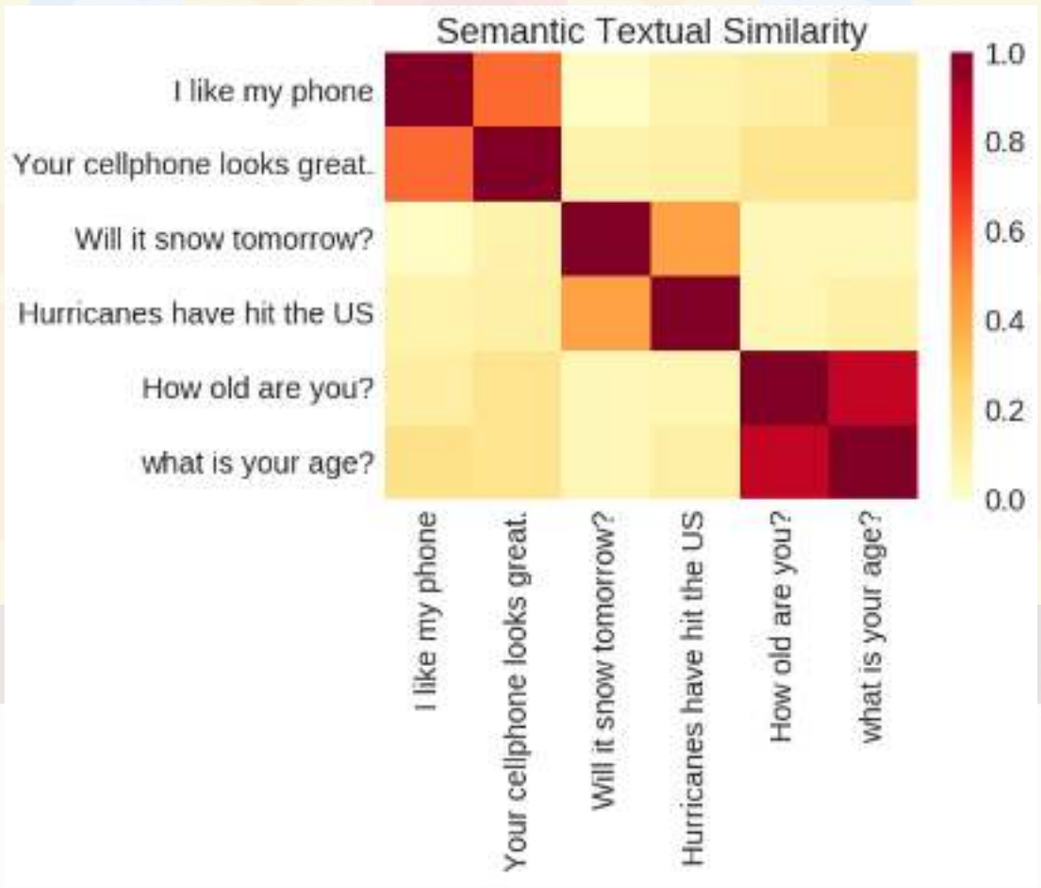
Cavaet on using USE

- One big caveat: USE only knows what it's trained on
  - Ex.: Feeding the same USE algorithm WSJ text

> Samsung Electronics Co., suffering a handset sales slide, revealed a foldable-screen smartphone that folds like a book and opens up to tablet size. Ah, horror? I play Thee to her alone;
> And when we have withdrom him, good all.
> Come, go with no less through.
>
> Enter Don Pedres. A flourish and my money. I will tarry.
> Well, you do!
>
> LADY CAPULET.
> Farewell; and you are

How does USE work?

- USE is based on a DAN
  - There is another specification as well
  - Learns the meaning of sentences via words' meanings
- Learn more: Original paper and TensorFlow site
- In practice, it works quite well



Try it out!

- Run on Google Colab
  - Python code
  - Just click the cells in order, and click run
  - Colab provides free servers to run the code on
    - It still takes a few minutes to run though

Text data

Other methods with text

- Vector space models are very common for text, but there are other methods:
  - LSTM for text generation or comprehension
    - Or RNN when using short snippets
  - LSTM can also be used for *translation*
  - CNN can be used on text
  - GAN or VAE can be used for text generation

LSTM for translation

- Seq2seq is a method for converting a **sequence** to a **sequence**
  - It creates a hidden sequence to facilitate translation
  - It comprises 2 neural networks:
    1. An LSTM from input to the hidden sequence
    2. An LSTM from the hidden sequence to the output

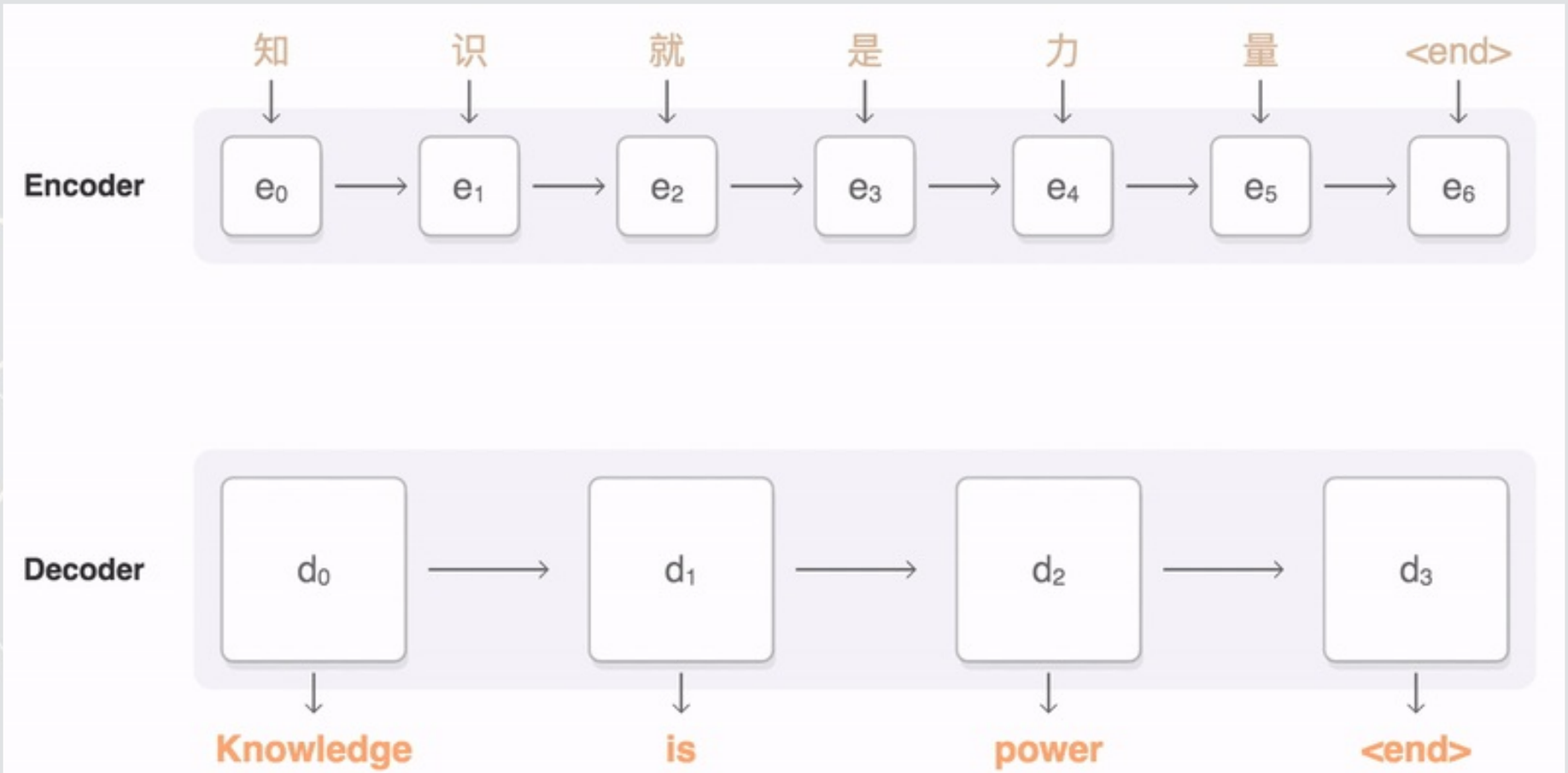


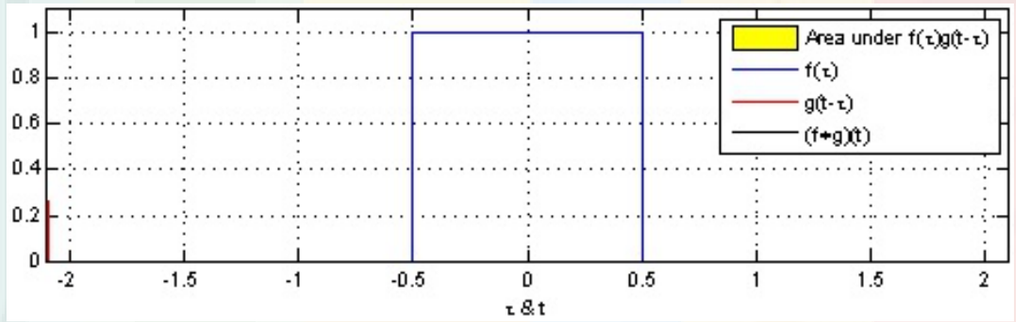
Image data

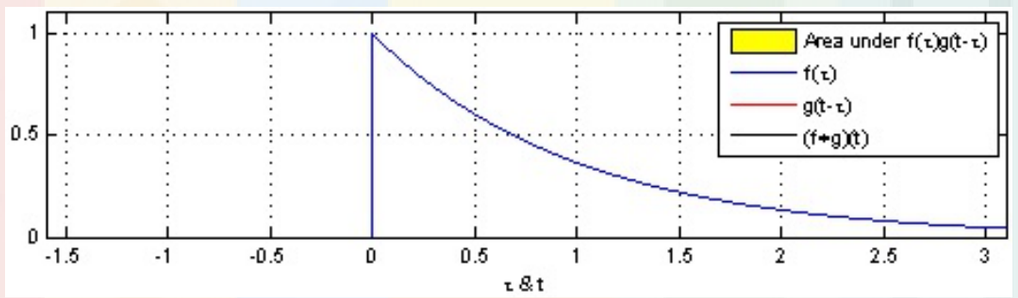
Try it out!

- Fashion MNIST with Keras and TPUs
  - Fashion MNIST: A dataset of clothing pictures
  - Keras: An easier API for TensorFlow
  - TPU: A “Tensor Processing Unit” – A custom processor built by Google

How CNNs work

- CNNs use repeated convolution, usually looking at slightly bigger chunks of data each iteration
- But what is convolution? It is illustrated by the following graphs (from Wikipedia):





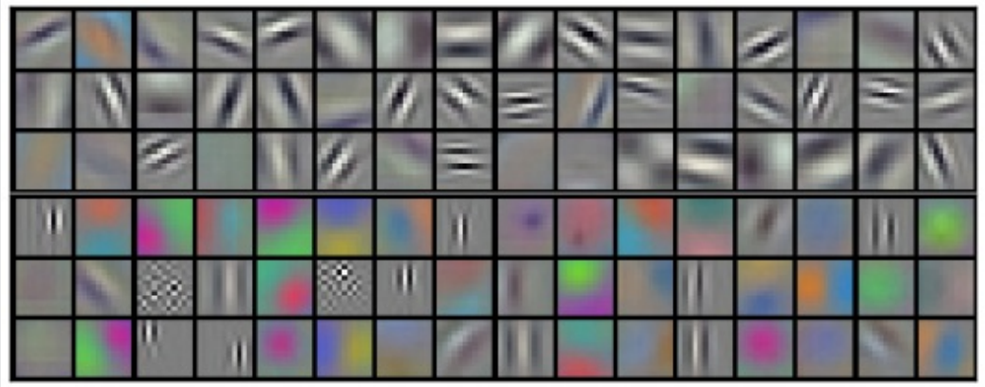
Further reading

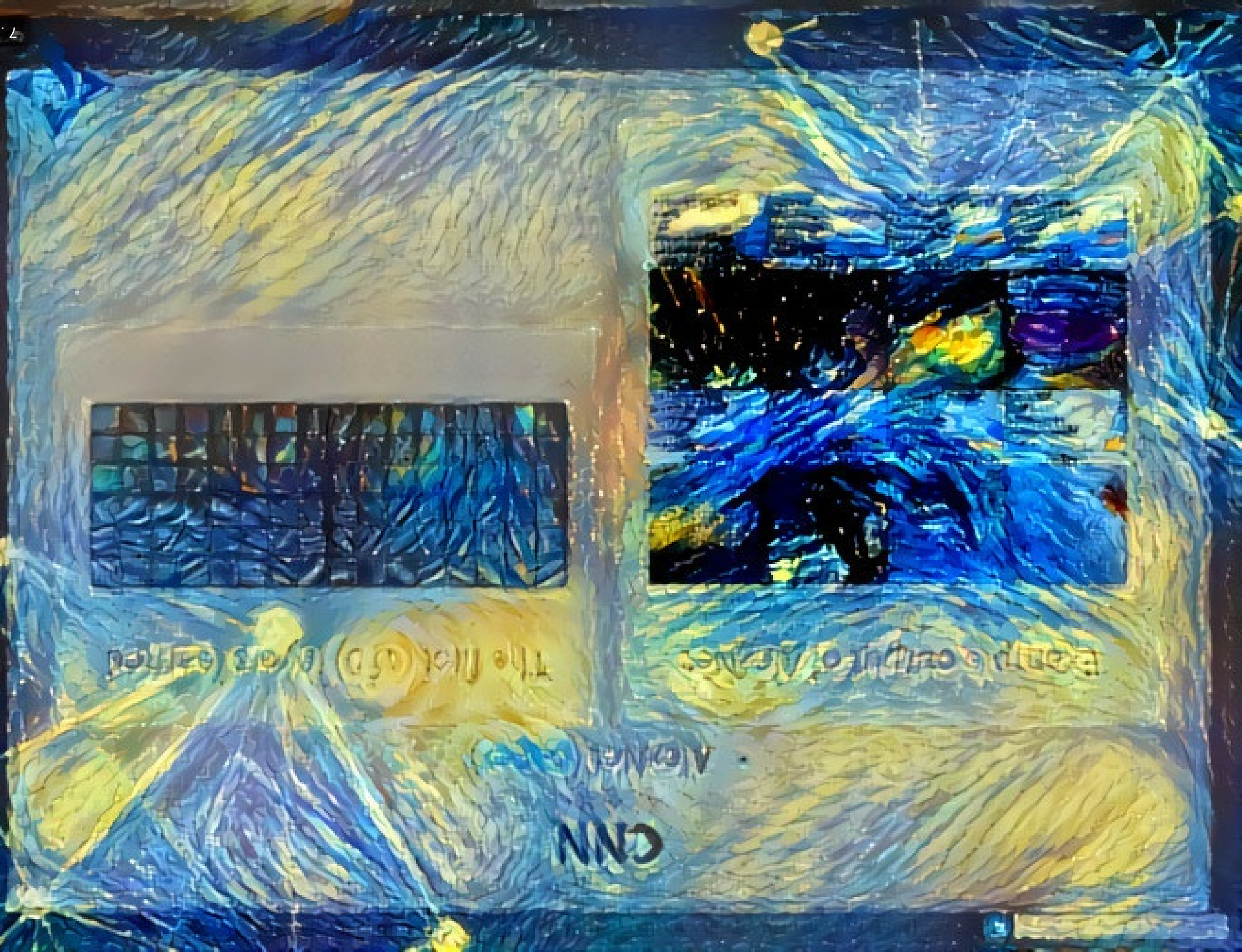
CNN

- AlexNet (paper)

Example output of AlexNet

The first (of 5) layers learned





CNN

AlexNet (2012)

Example output of AlexNet

The first (of 5) layers learned

Transfer Learning

- The previous slide is an example of *style transfer*
- This is also done using CNNs
- More details here

Try it out!

- Colab file available at this link
  - Largely based off of dsgiitr/Neural-Style-Transfer
  - It just took a few tweaks to get it working in a Google Colaboratory environment properly

Inputs:

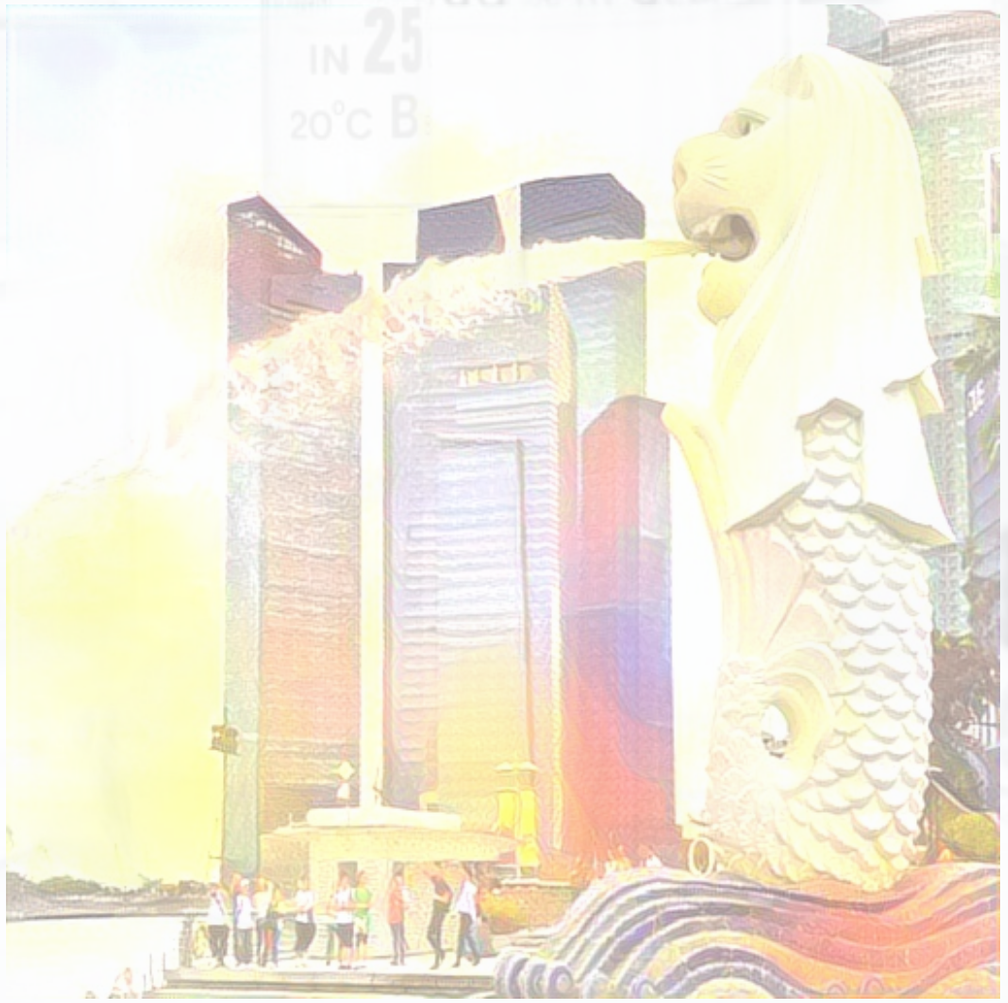
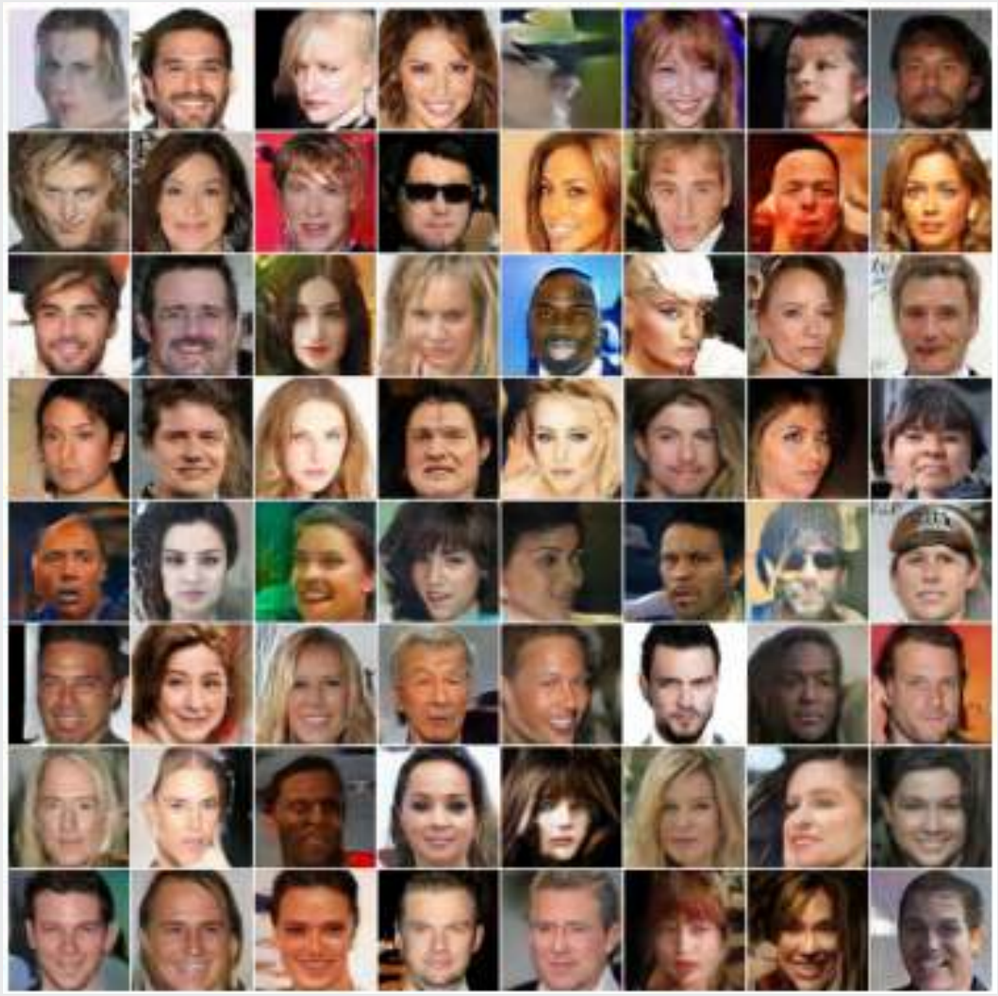


Image generation with VAE

- Example from yzwxx/vae-celeb

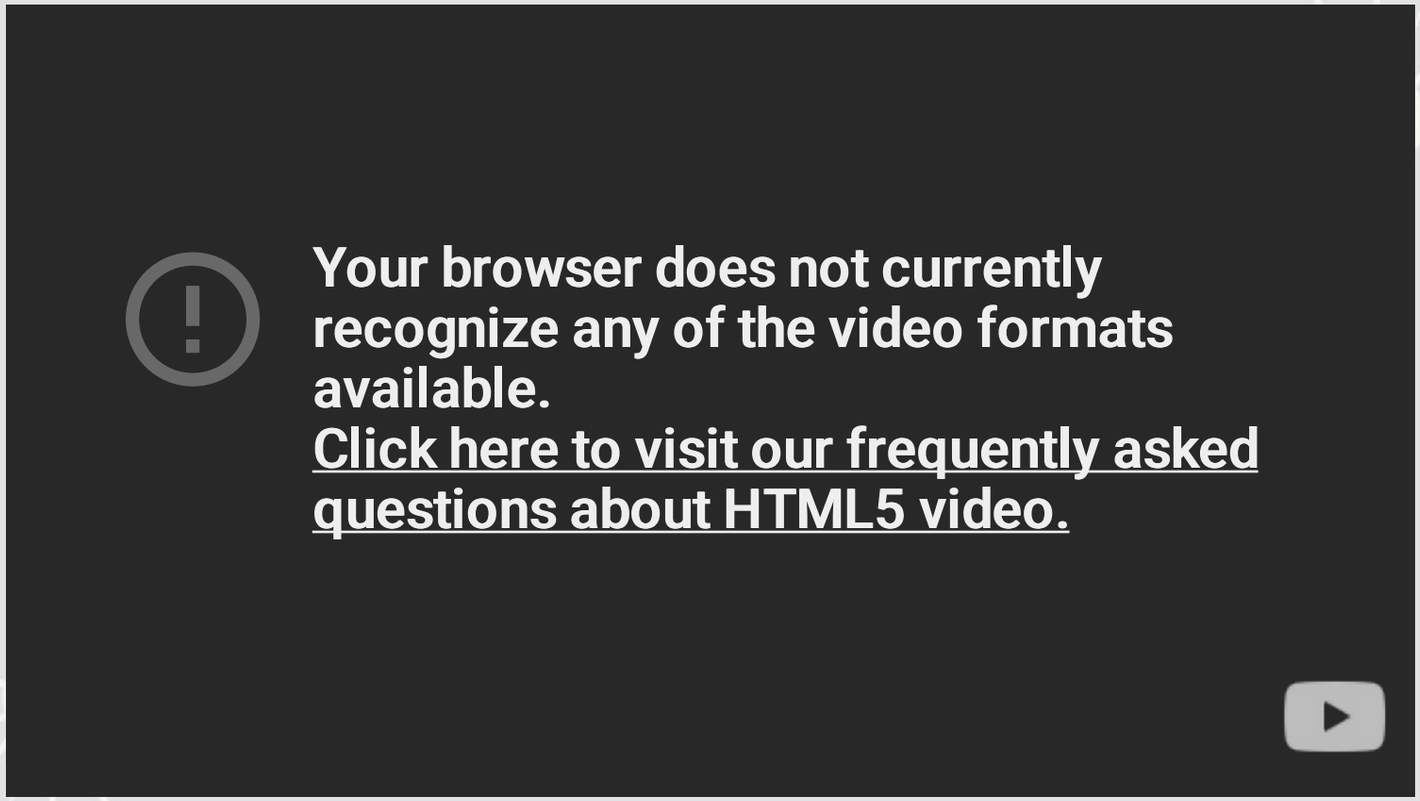
Input and autoencoder

Generated celebrity images



Note on VAE

- VAE doesn't just work with image data
- It can also handle sound, such as MusicVAE



Code for trying on your own

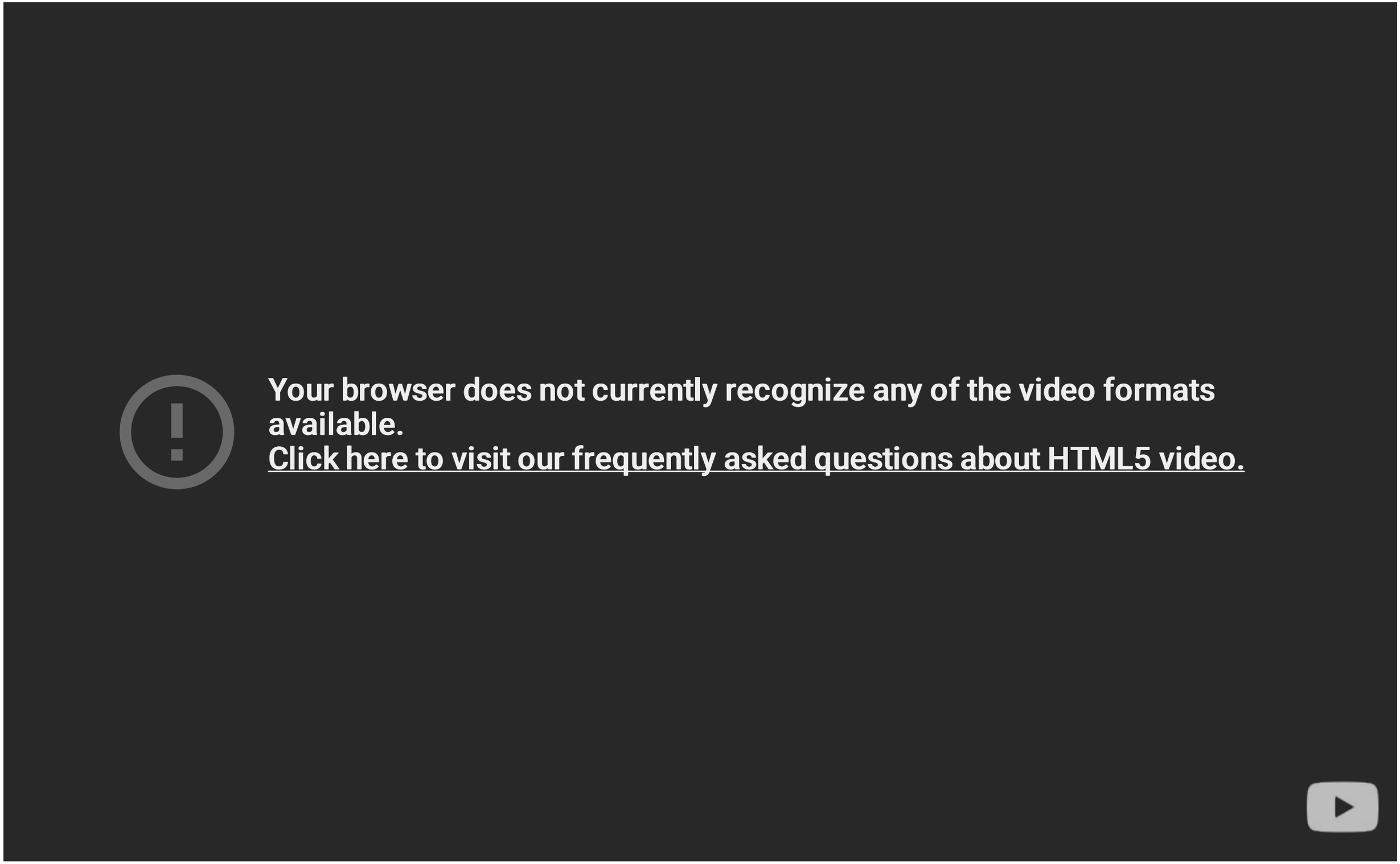
Video data

One method for video

YOLOv3

- You
- Only
-

- Once


Your browser does not currently recognize any of the video formats available.
Click here to visit our frequently asked questions about HTML5 video.

Video link

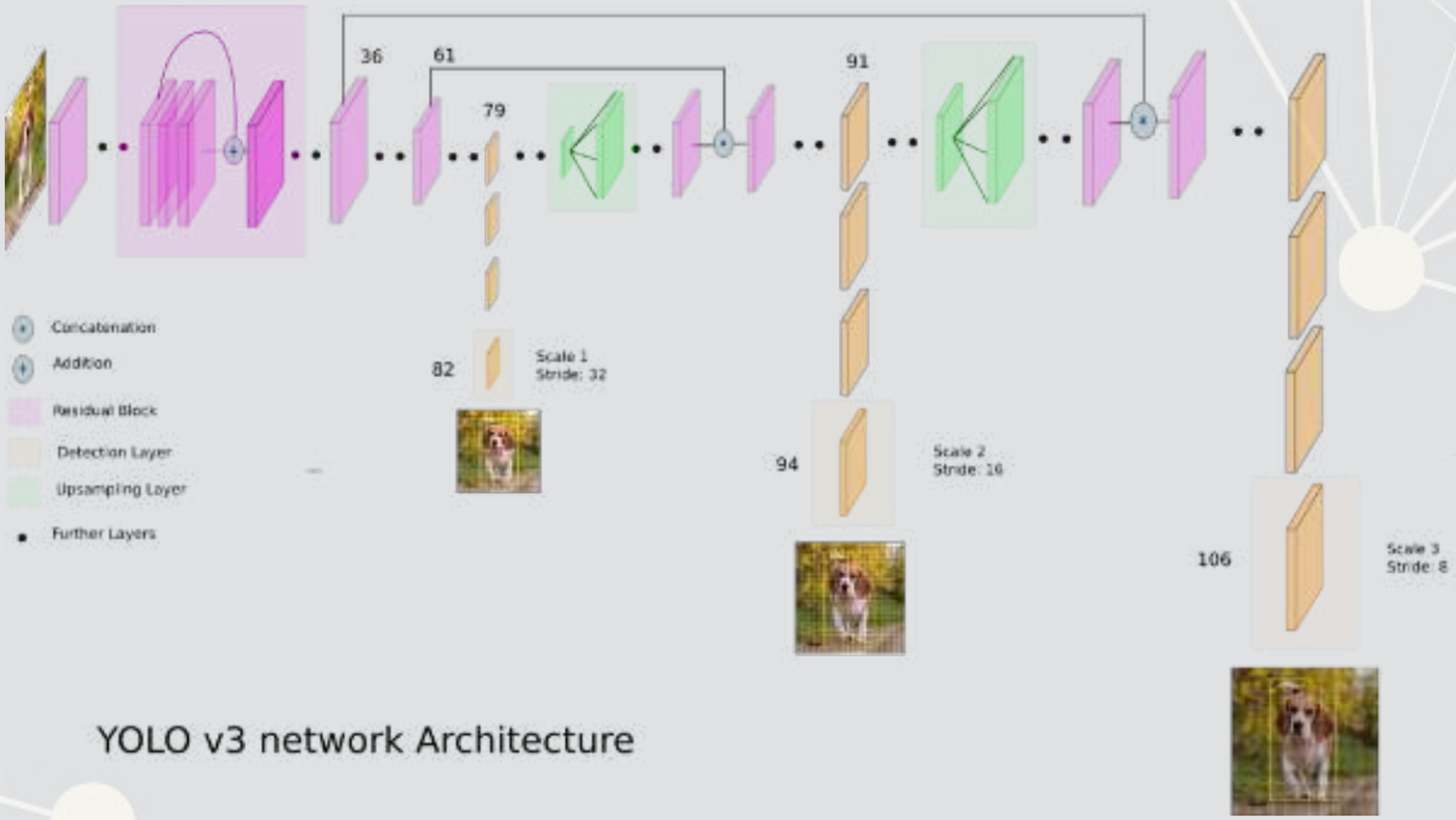
What does YOLO do?

- It spots objects in videos and labels them
  - It also figures out a *bounding box* – a box containing the object inside the video frame
- It can spot *overlapping* objects
- It can spot multiple of the same or different object types
- The baseline model (using the COCO dataset) can detect 80 different object types
  - There are other datasets with more objects

How does Yolo do it? Map of Tiny YOLO

> Yolo model and graphing tool from lutzroeder/netron

How does Yolo do it?



YOLO v3 network Architecture

Diagram from *What's new in YOLO v3* by Ayoosh Kathuria

Final word on object detection

- An algorithm like YOLO v3 is somewhat tricky to run
- Preparing the algorithm takes a long time
  - The final output, though, can run on much cheaper hardware
- These algorithms just recently became feasible
  - So their impact has yet to be felt so strongly

> Think about how facial recognition showed up everywhere for images over the past few years

End matter

Final discussion

> What creative uses for the techniques discussed today do you expect to see become reality in accounting in the next 3-5 years?

- 1 example: Using image recognition techniques, warehouse counting for audit can be automated
  - Strap a camera to a drone, have it fly all over the warehouse, and process the video to get item counts

Recap

Today, we:

- Learned formally what neural networks (NNs) are
- Discussed a variety of NN-based algorithms
  - And observed various applications of them

For next week

- For next week:
  - Finish the group project!
    1. Kaggle submission closes tomorrow night!
       - At least for the non-Google groups
    2. Turn in your code and report through eLearn's dropbox
    3. Prepare a short (12-15 minute) presentation for class

More fun examples

- Interactive:
  - Performance RNN
  - TensorFlow.js examples
- Others:
  - Google’s deepdream
  - Open NSynth Super

Fun machine learning examples

- Interactive:
  - Semantris
    - A game based on the Universal Sentence Encoder
  - Draw together with a neural network
    - click the images to try it out yourself!
  - Google's Quickdraw
  - Google's Teachable Machine
  - Four experiments in handwriting with a neural network
- Non-interactive
  - Predicting e-sports winners with Machine Learning

> For more reading, see the **gifts** on eLearn

Packages used for these slides

- `kableExtra`
- `knitr`
- `tidyverse`
  - `dplyr`, `magrittr`, `readr`

Generating Shakespeare

```
seed_txt = 'Looks it not like the king?  Verily, we must go! '  # Original code
seed_txt = 'SCENE I. Elsinore. A platform before the Castle.\n\n Enter Francisco a
seed_txt = 'Samsung Electronics Co., suffering a handset sales slide, revealed a f
# From: https://www.wsj.com/articles/samsung-unveils-foldable-screen-smartphone-15
```