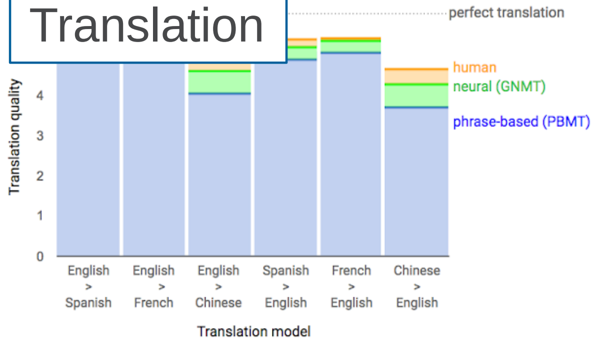
A MINIMAL INTRODUCTION TO NEURAL NETWORKS

LOGAN WARD
Asst. Computational Scientists
Data Science and Learning Division

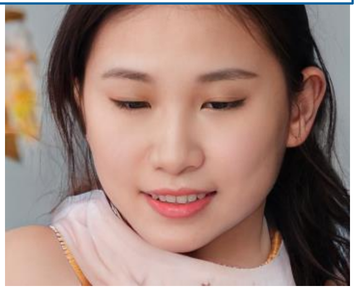
NEURAL NETWORKS ARE *VERY* VERSATILE

Translation



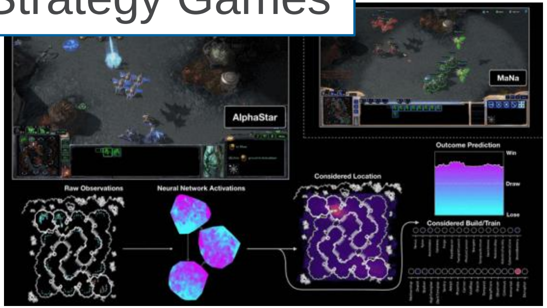
Source: Google Blog

Image Generation



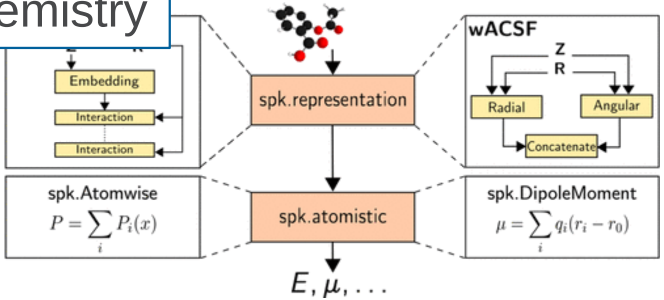
Source: https://thispersondoesnotexist.com/

Strategy Games



Source: OpenAI

Chemistry



Source: Schutt et al. JCTC. (2019)

GOALS FOR TODAY

Get familiar enough to start using neural networks

Goal 1: Describe the three key components of neural networks

Goal 2: Train a neural network with TensorFlow

Goal 3: Understand why CNN are used for images

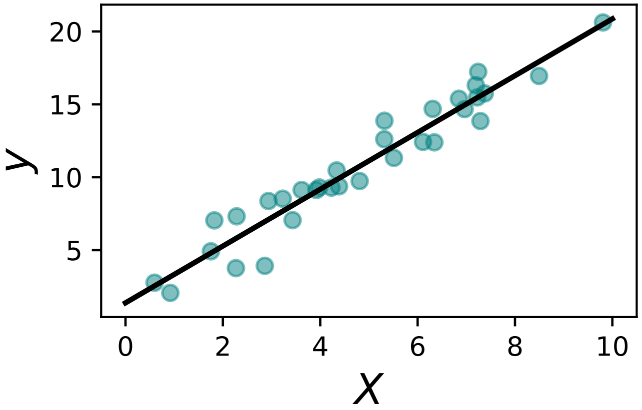
CONCEPT OF NEURAL NETWORKS: ARCHITECTURE + LOSS FUNCTION + SOLVER

AN OLD FRIEND: SIMPLE LINEAR REGRESSION

… but let's make it sound modern

Model Architecture

$$\boldsymbol{f(x; m, b) = mx + b}$$



Training Data: Inputs (x_i) and outputs (y_i)

Goal: Determine m and b that minimize

Loss Function

$$\sum_i (f(x_i; m, b) - y_i)^2$$

by computing

Optimizer

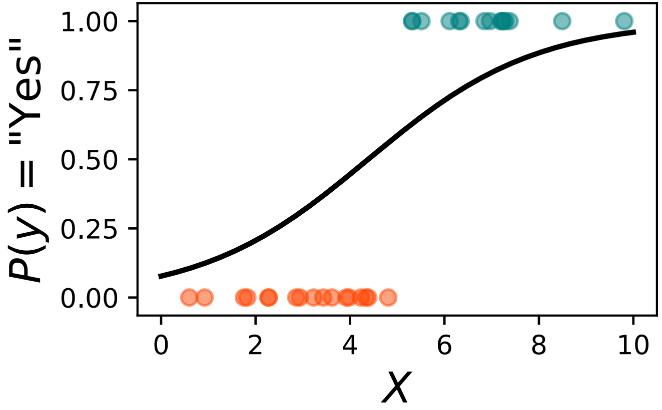
$$m = \mathrm{Cov}[x, y]/Var[x]$$
$$b = \bar{y} - m\bar{\mathrm{x}}$$

SIMPLE LOGISTIC REGRESSION

A version of Linear Regression suitable for classification

Model Architecture

$$\boldsymbol{f(x; m, b) = \frac{1}{1+e^{-(mx+b)}}}$$



Training Data: Inputs (x_i) and outputs (y_i)

Goal: Determine m and b that minimize

Loss Function

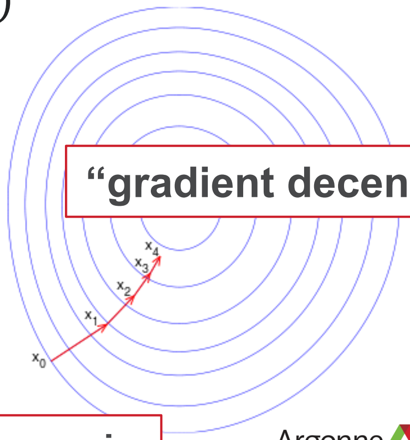
$$L(m,b)\sum_i y_i \ln\big(f(x_i)\big) + (1-y_i)\ln(1-f(x_i))$$

"log loss"*

by computing

Optimizer

$$x_0 = (1,0)$$
$$x_{n+1} = x_n + \gamma\nabla L(m,b)$$



"gradient decent"*

Architecture + Loss + Optimizer = ML Algorithm
For Regression *and* Classification

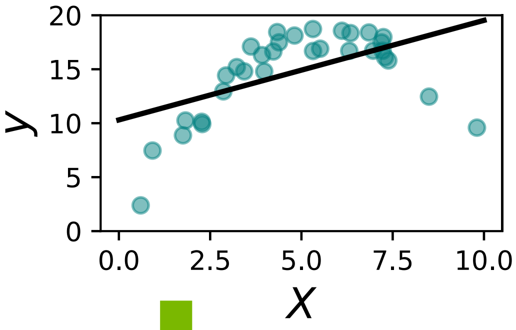
***You will be seeing these again**

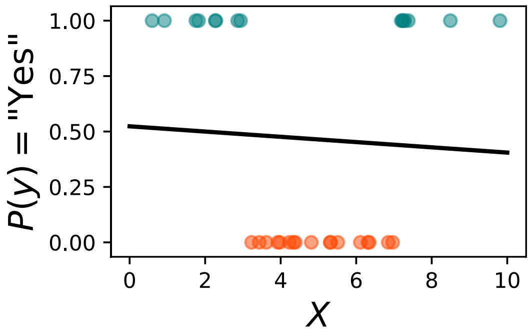
LINEAR MODELS ARE NOT SUFFICIENT

Otherwise, this would be a very short lecture

Why Not? Model complexity is limited

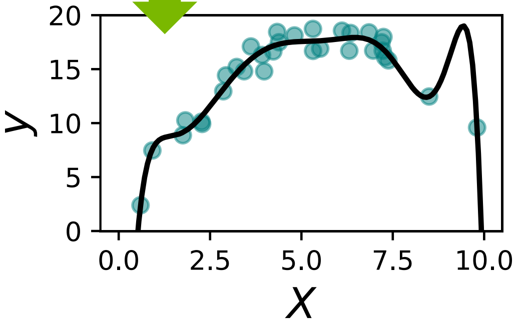
"underfit"*





… and adding complexity comes with risks

"overfit"*



Key Questions for Neural Networks:

1. How to add more complexity?
2. How to limit "overfitting"?

NEURAL NETWORKS ARE *COMPOSABLE, NON-LINEAR* MODELS

TODAY'S FOCUS: NEURAL NETWORKS

Build on a small building block: "The Perceptron"

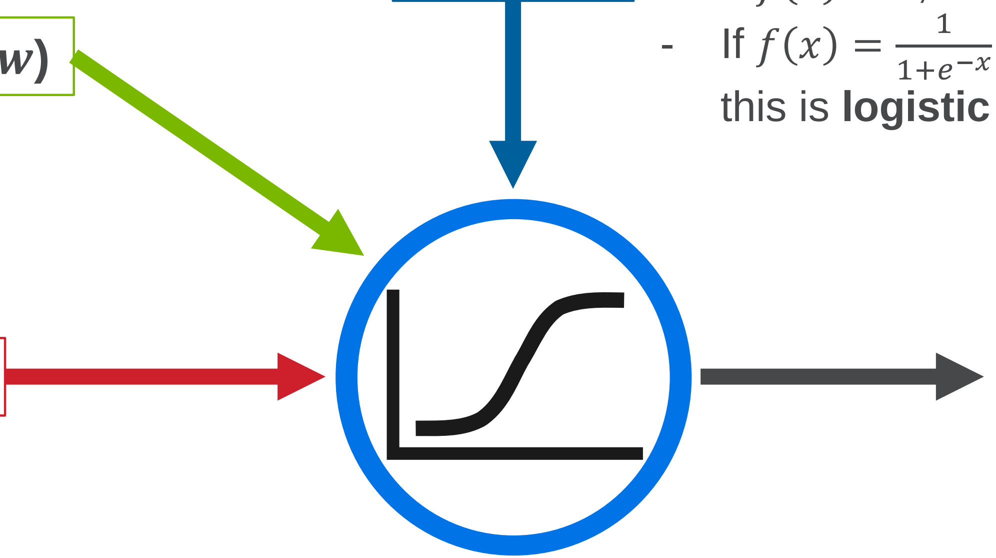
Notice that…

- If $f(x) = \mathrm{x}$, this is **linear regression**
- If $f(x) = \frac{1}{1+e^{-x}}$, this is **logistic regression**

Bias (b)

Weight (w)

Input (x)



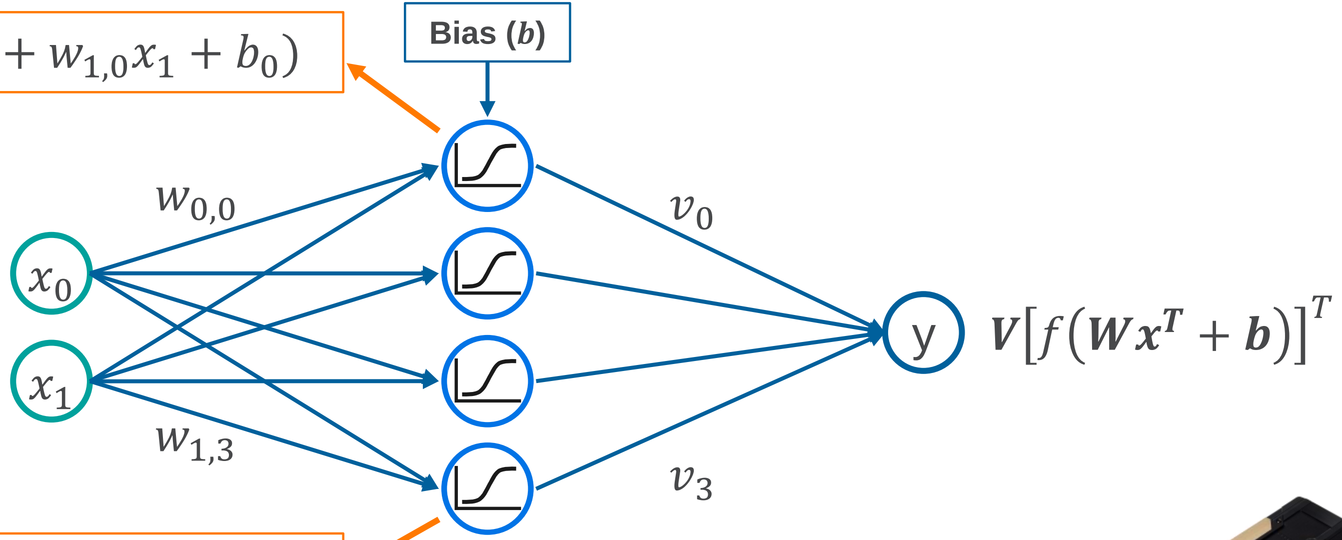
Output:
$$y = f(wx + b)$$

Activation Function (f)

MANY PERCEPTRONS = NEURAL NETWORK

Stack enough, and you get *very* complex functions

$$h_0 = f(w_{0,0}x_0 + w_{1,0}x_1 + b_0)$$



$$h_3 = f(w_{0,3}x_0 + w_{1,3}x_1 + b_3)$$

Why popular now?

Many small operations
<u>+ performed on many data</u>
Massive Parallelism

HOW DO I TRAIN A NEURAL NETWORK?

Remember when I said “gradient decent”

Key Terminology:

Architecture: How inputs/outputs are linked, adjustable weights

Loss Function: Generates error between “current” and “desired” outputs

Optimizer: Algorithm for finding parameters that minimize a function

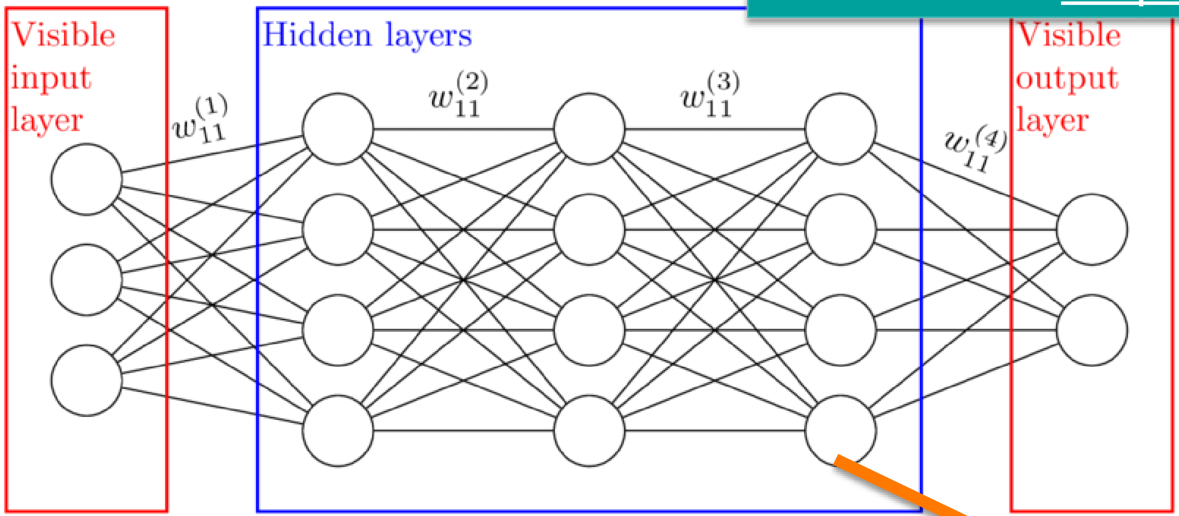
These are the three ingredients forming all* neural networks

*Probably an overgeneralization

NETWORKS ARE COMPOSED OF LAYERS

Tensors in, different tensors out

Complicated networks break down into simple layers



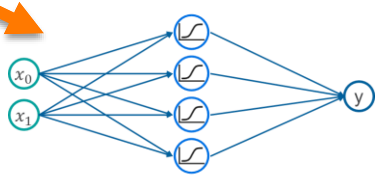


Figure: DeepAI.org

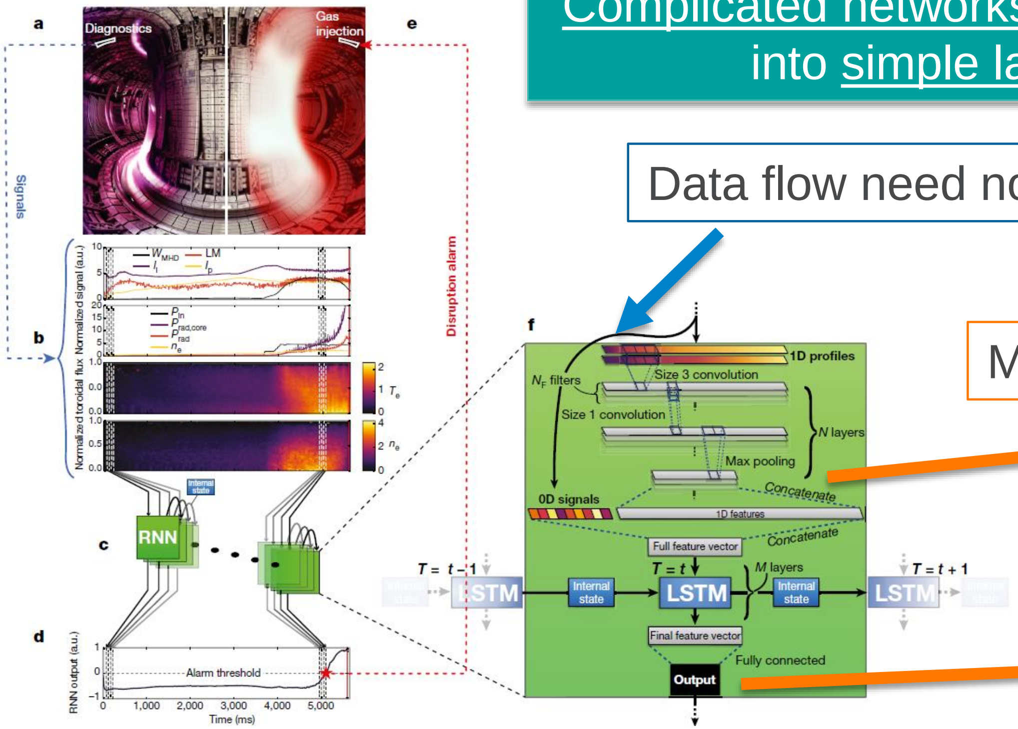
NETWORKS ARE COMPOSED OF LAYERS

Tensors in, different tensors out

Complicated networks break down into simple layers

Data flow need not be sequential

Mix-and-match computations




```
y = x.ravel()
```

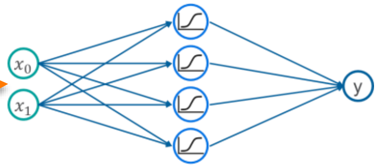


Figure: Keats-Harbeck et al. Nature. 2019

LOSS FUNCTIONS: NOT JUST "LOG LOSS"

Express how "wrong" your network is as a differentiable function

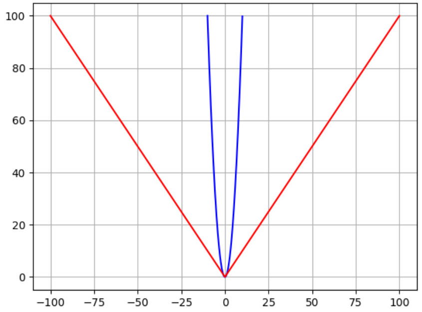


Figure: Towards Data Science

Regression:

Mean Absolute Error: $L = \sum_i |\hat{y}_i - y_i|$

Mean Squared Error: $L = \sum_i (\hat{y}_i - y_i)^2$

Classification:

~~**Accuracy:**~~ Not differentiable!

Log Loss: $L = \sum_i \sum_c (\boldsymbol{y_i} = \boldsymbol{c}) \log P(y_i = c)$

Only counts for the correct class

Bigger penalty if more wrong

LOSS FUNCTIONS: NOT JUST "LOG LOSS"

Express how "wrong" your network is as a differentiable function

Mean Squared Error: Standard for regression problems

Huber Loss: Less outlier-sensitive than MSE

Jaccard Loss: Used for image bounding boxes

https://www.pyimagesearch.com/2016/11/07/intersection-over-union-iou-for-object-detection/

"KL" Divergence: Ensure outputs follow a desired distribution

Loss gives another tool to control training

HOW DO I TRAIN A NETWORK?

Short Answer: Gradually make the weights better

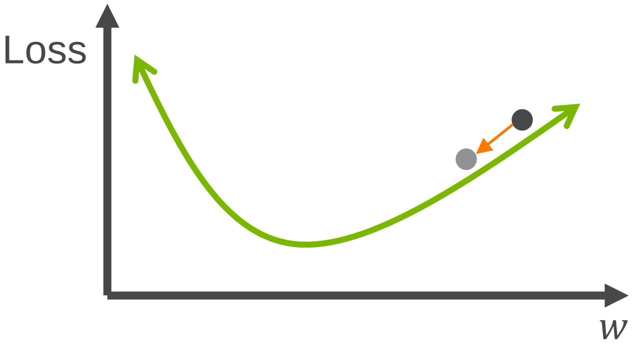
Simple procedure:

1. Compute output
2. Compute "loss"
3. Compute how each weight affects loss
   (Uses "back propagation")
4. Adjust weights to lower loss
   (More complicated than you might think)
5. Repeat with new weights

$$\hat{y} = f(X; w)$$

$$L = (y - \hat{y})^2$$

$$w' = w + \gamma \frac{\delta L}{\delta w}$$



THERE IS A RICH VARIETY IN NEURAL NETWORKS

Optimizers, layers, and loss functions

Layers

Activation: Applies function to an input

Batch Normalization: Make batch mean 0, std. 1

Convolution: Apply spatial/temporal filters

… Dense, Dropout, Embedding, ….

Loss Functions

Log-loss: Classification, same loss function as logistic regression

Mean Absolute Error: Regression, <u>small penalty</u> for outliers

Mean Squared Error: Regression, <u>large penalty</u> for outliers

… KL divergence, accuracy …

Optimizers

Many different techniques:

Momentum: Keep moving in direction of last step

Decay: Gradually lower step size

Clipping: Prevent too large of gradient changes

THERE IS A RICH VARIETY IN NEURAL NETWORKS

Optimizers, layers, and loss functions

Layers

Activation: Applies function to an input

Batch Normalization: Make batch mean 0, std. 1

Convolution: Apply spatial/temporal filters

… Dense, Dropout, Embedding, ….

Loss Functions

Log-loss: Classification … gistic regression

Mea … sion, small penalty for outliers

… **uared Error:** Regression, large penalty for outliers

… KL divergence, accuracy …

Learning when to apply what technique takes practice!

Optimizers

Many different techniques:

Momentum: Keep moving in direction of last step

Decay: Gradually lower step size

Clipping: Prevent too large of gradient changes

DNN EXERCISE: KEY SKILLS

Learning how to make and train a model effectively with Keras

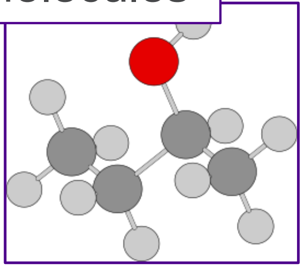
- Open the first exercise!

WHAT IF MY DATA ISN'T A VECTOR!?

NOT ALL DATA ARE VECTORS

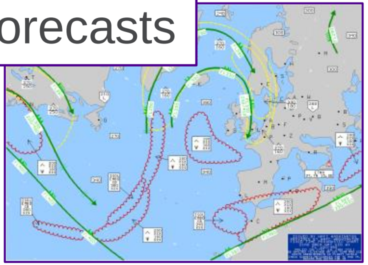
And that's OK!

Molecules

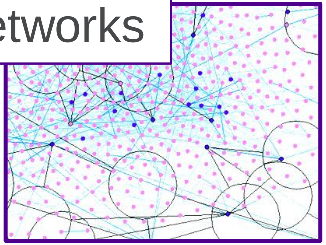


Images

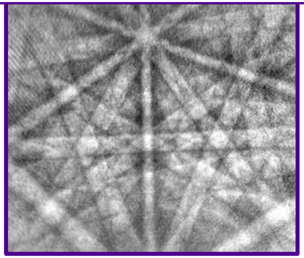
Weather Forecasts



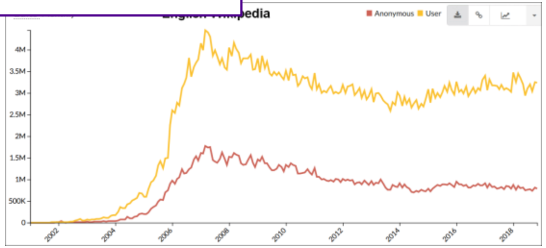
Social Networks



EBSD Patterns



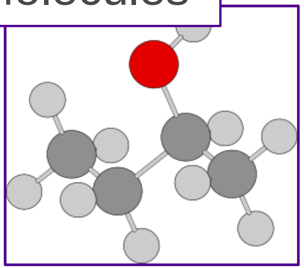
Timeseries



NOT ALL DATA ARE VECTORS

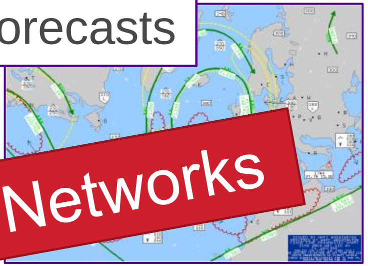
And that's OK!

Molecules



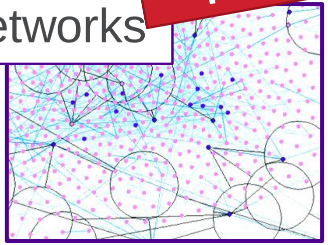
Images

Weather Forecasts

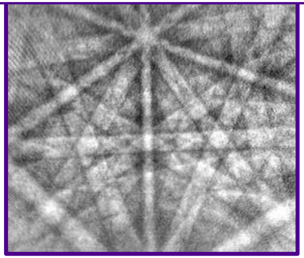


Special Data Can Need Specialized Networks

Social Networks



EBSD Patterns



Timeseries

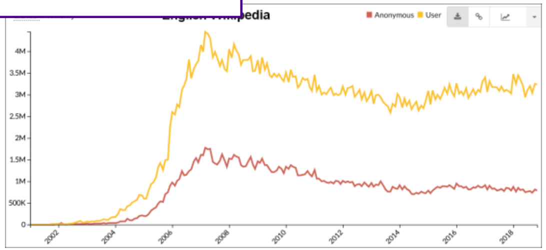
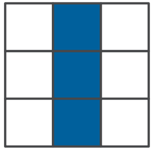
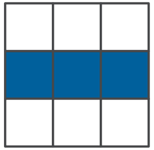


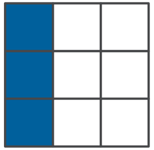
IMAGE CLASSIFICATION AND CONVOLUTIONS

Better classification by translation symmetry

Example: Classify Horizontal vs Vertical lines







Initial Approach: Just flatten the images. They are now vectors.

How do we know which are which? Adjacent blue blocks

Problem! Fully connected NNs don't care about order

Solution: Make new features that deal with order

CONVOLUTIONS, PADDING, AND POOLING

Borrow from computer vision, graphics

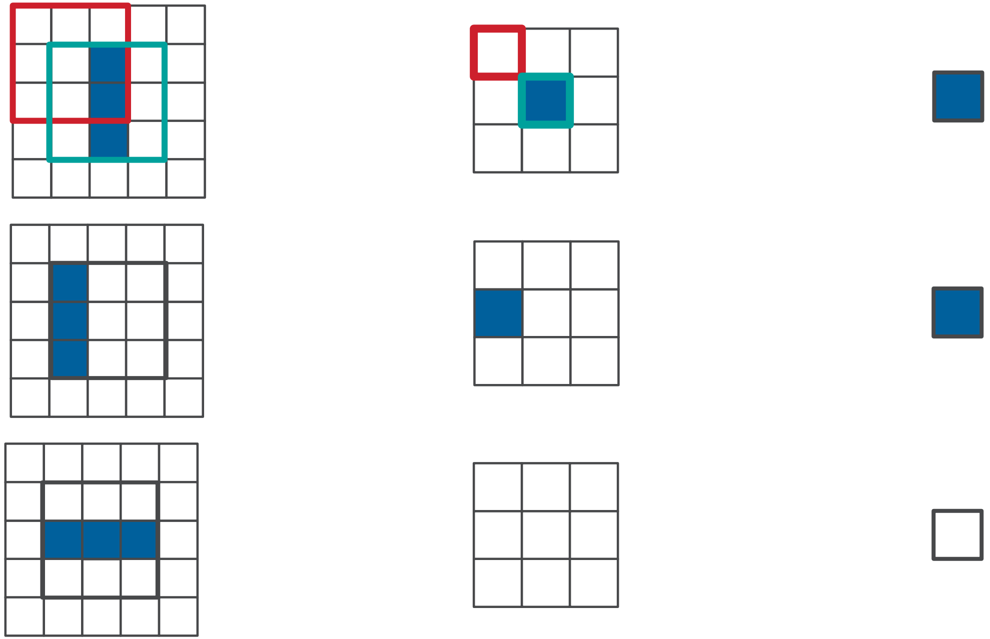
1. Pad Image

2. Convolve Filter

3. Maximum of Image ("Pooling")

Vertical Edge Filter:

0	1	0
0	0	0
0	1	0



Classification is **easy** with filters!

CONVOLUTIONS, PADDING, AND POOLING

Borrow from computer vision, graphics

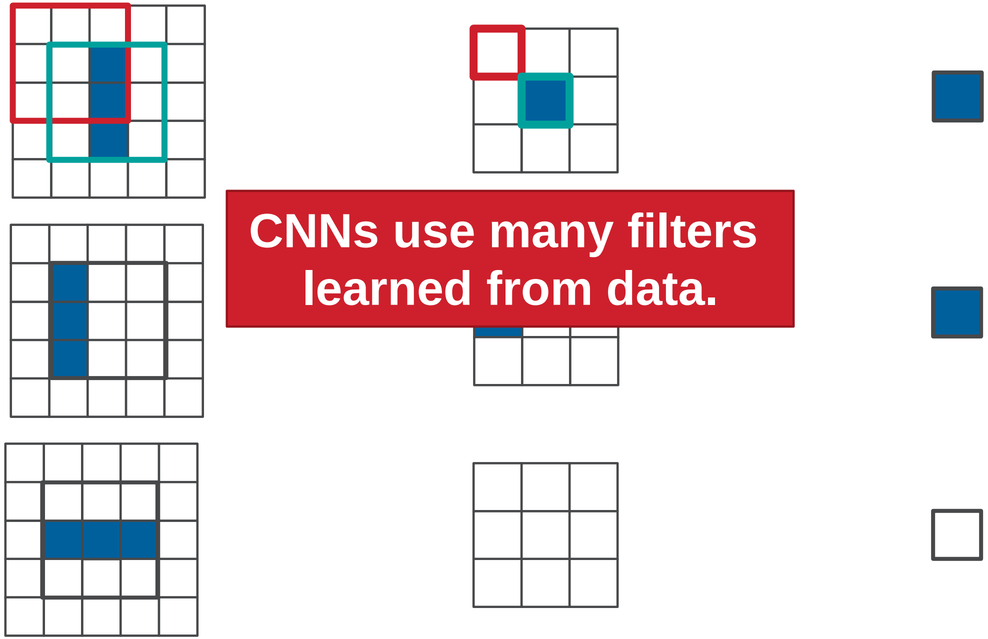
1. Pad Image

2. Convolve Filter

3. Maximum of Image ("Pooling")

Vertical Edge Filter:

0	1	0
0	0	0
0	1	0



CNNs use many filters learned from data.

Classification is **easy** with filters!

EXERCISE: DIGIT CLASSIFICATION

It’s a great tutorial example!

- Let’s do example #2

TAKE-HOME MESSAGES

1. **Neural Networks have three main components**
    1. *Architecture:* How the “perceptrons” are arranged
    2. *Loss Function:* Measures difference between “actual” and “expected”
    3. *Optimizer*: How network weights are adjusted to lower loss

2. **TensorFlow+Keras makes deep learning easy**
    – Compose layers to form network architectures
    – Use callbacks to prevent overfitting
    – Control batch size to improve efficiency

3. **Special data requires special networks**
    – General concept: Exploit symmetries / domain knowledge
    – Special Example: Convolutions exploit translation symmetry and that “nearby” pixels/inputs are related

EMAIL ME AT LWARD@ANL.GOV IF YOU HAVE QUESTIONS!