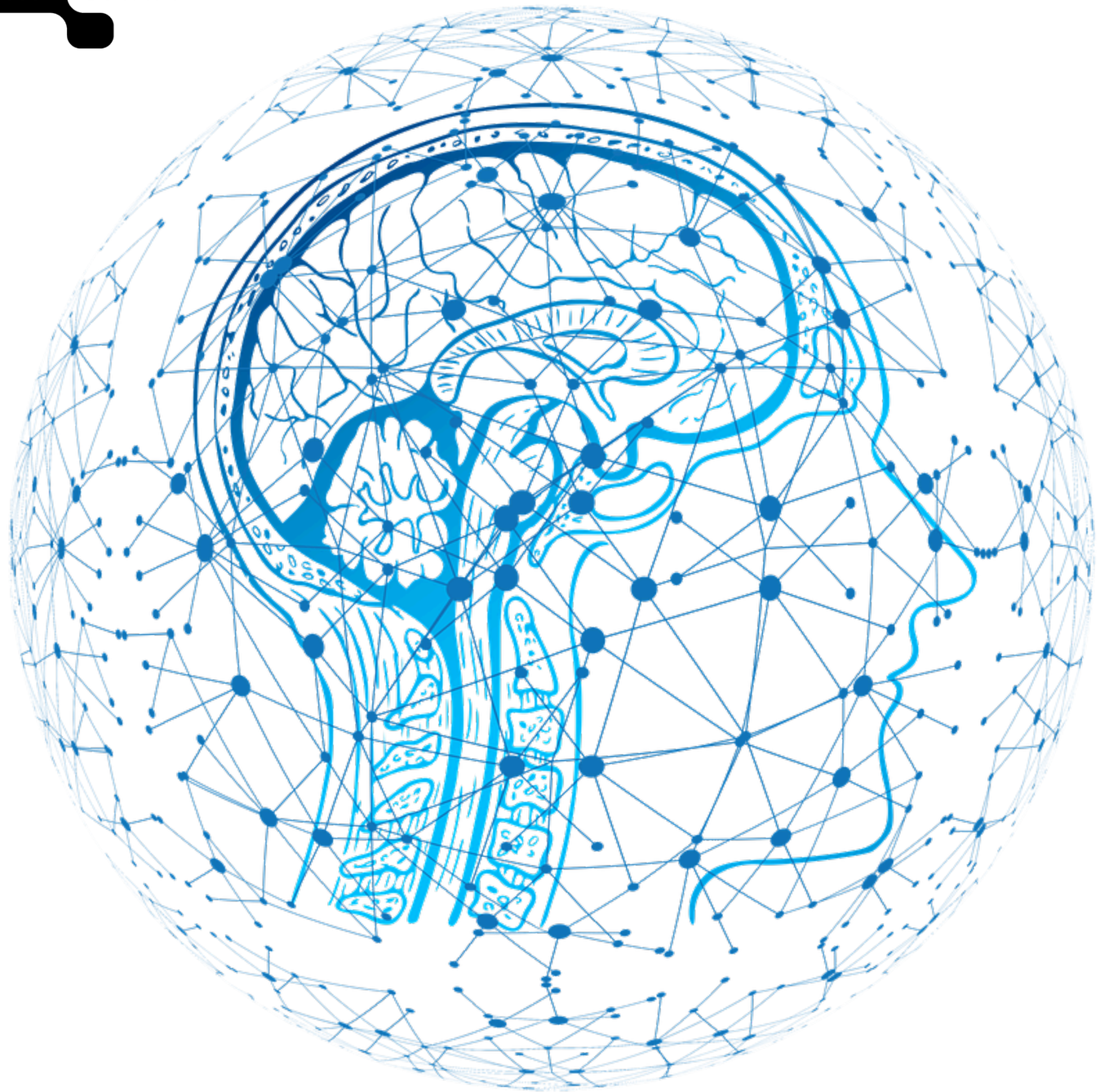
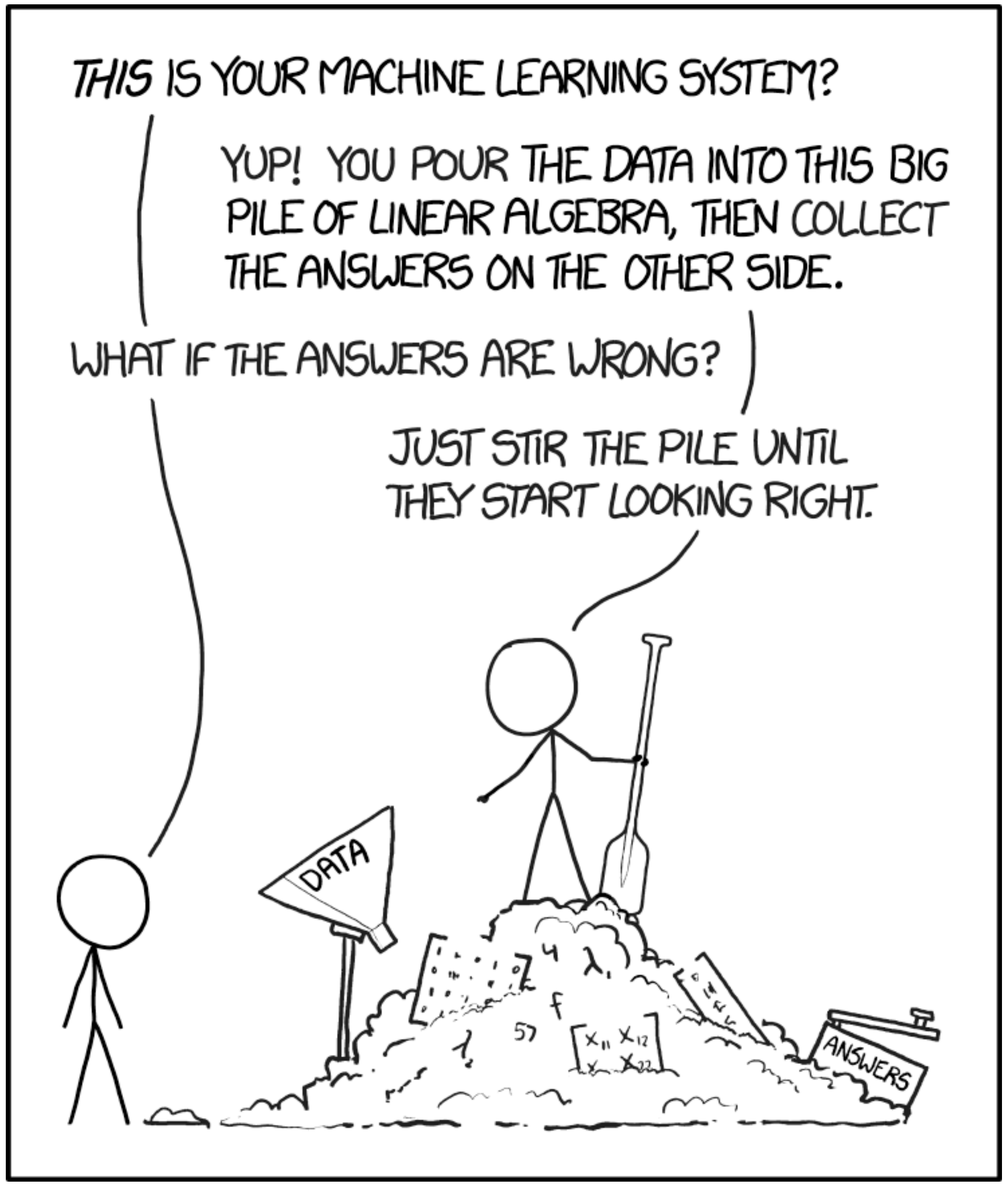


Introduction to High Performance Machine Learning

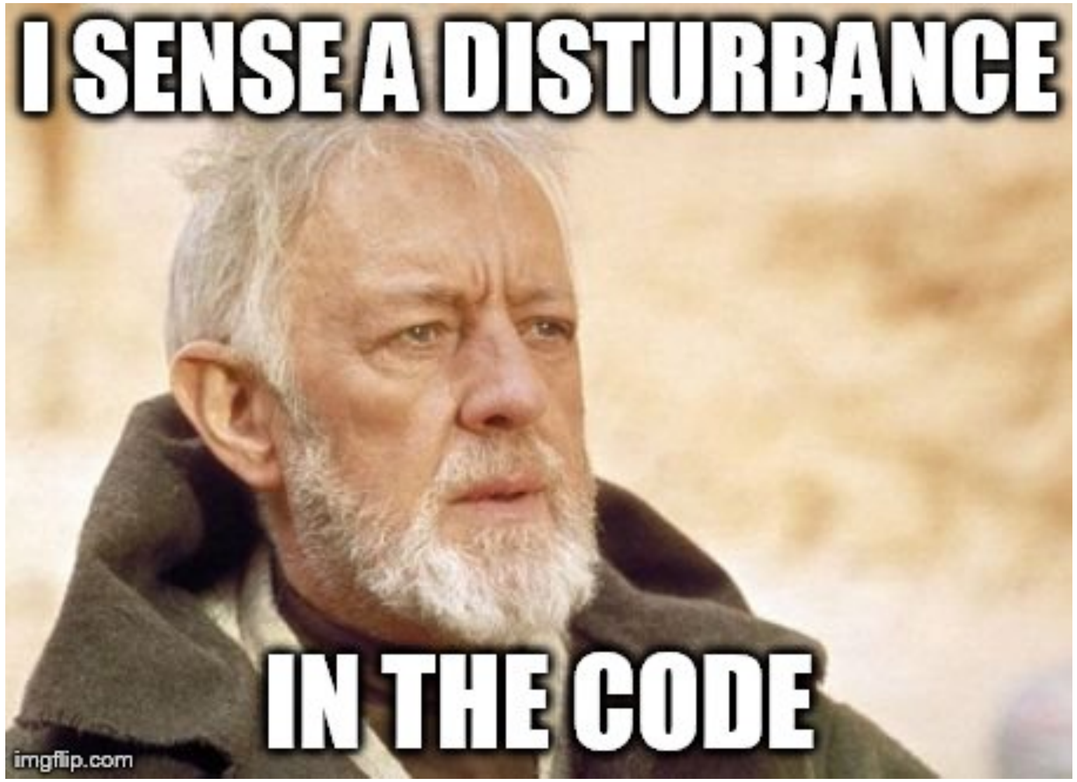
DNN Inspection
Result Interpretation

Maxwell Cai, PhD

Can we trust the results from the DNNs?



Source: xkcd



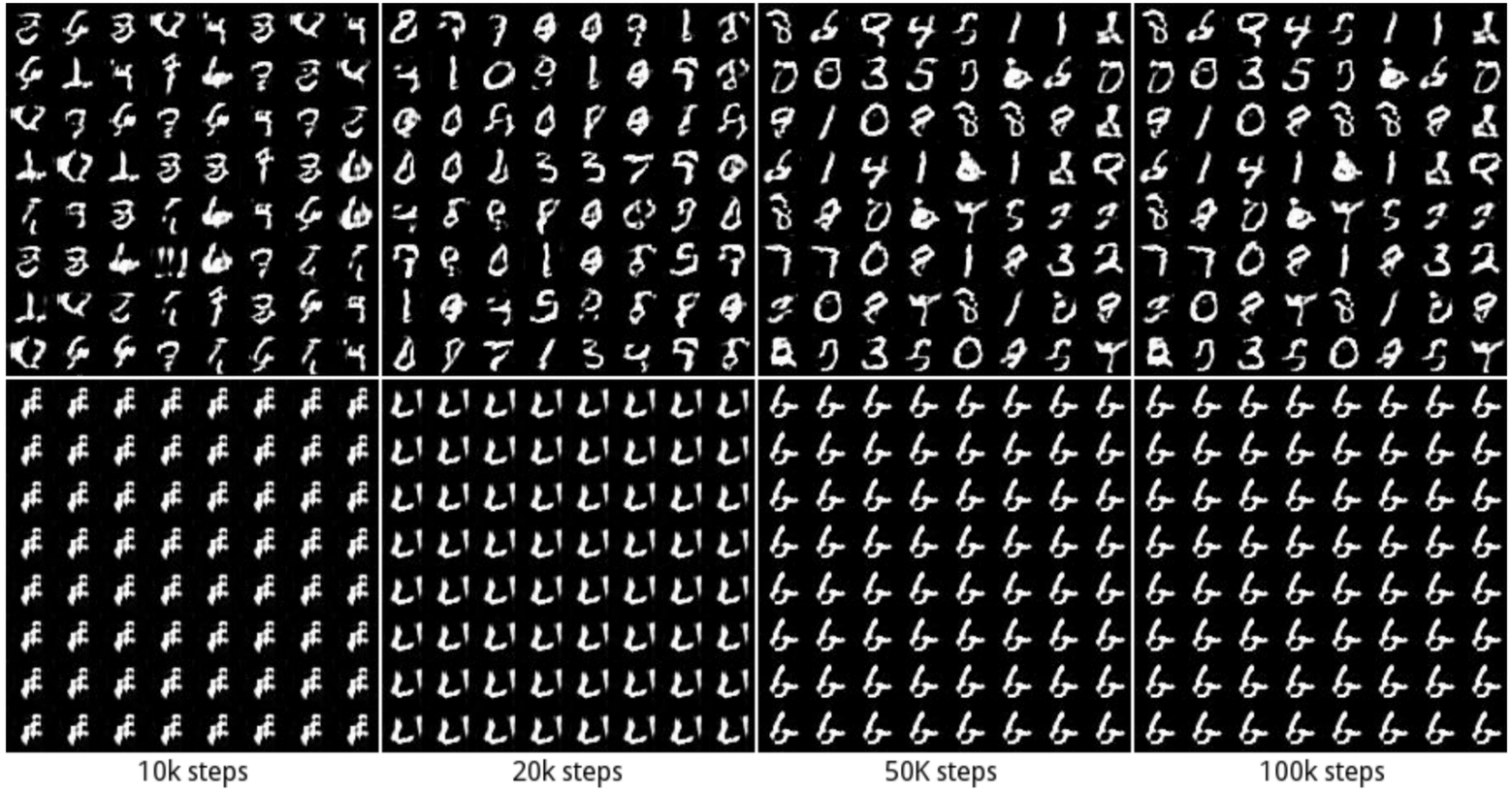
Challenges of debugging a DNN

Deep learning is a new programming paradigm driven by data. But…

- The code never crashes, raises an exception, or even slows down.
- The network still trains and the loss will still go down.
- The values converge after a few hours, but to really poor results

Metz et al. 2016 (arXiv:1611.02163)

Mode collapse of GANs

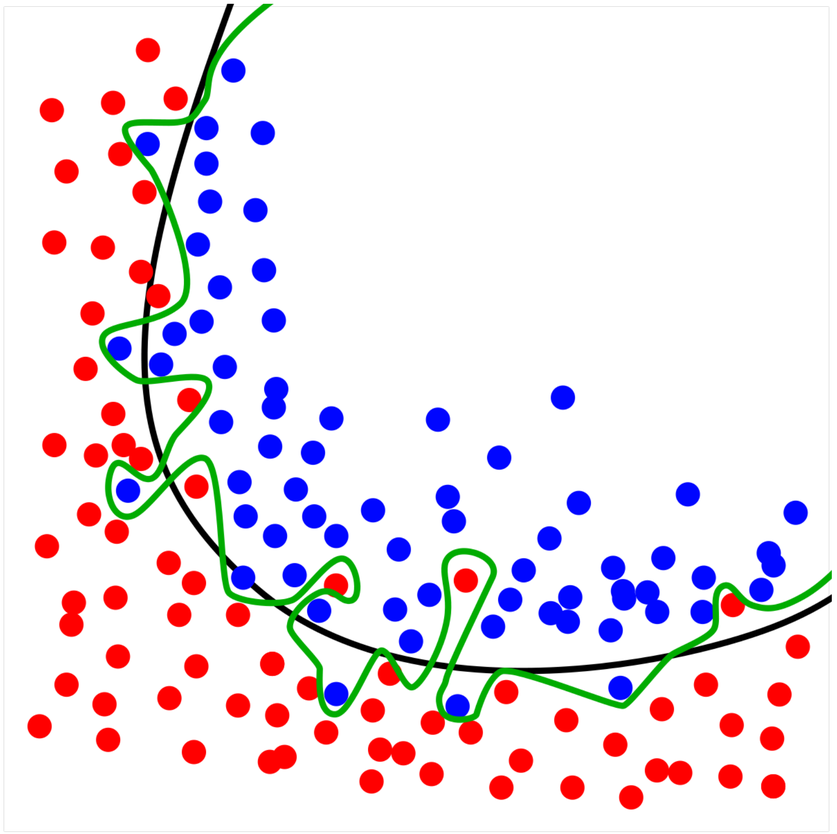


See also: https://medium.com/@keeper6928/how-to-unit-test-machine-learning-code-57cf6fd81765

What can we tell from the training/validation losses?

Overfitting

In case of overfitting, train/validation losses are helpful.

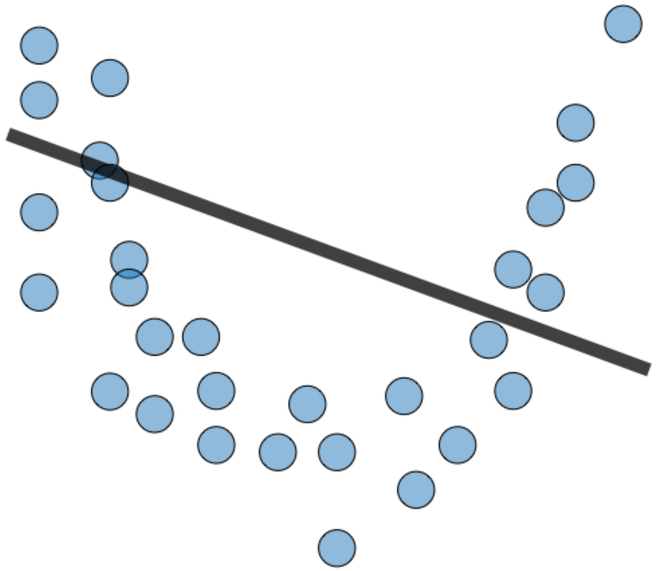


Underfitting

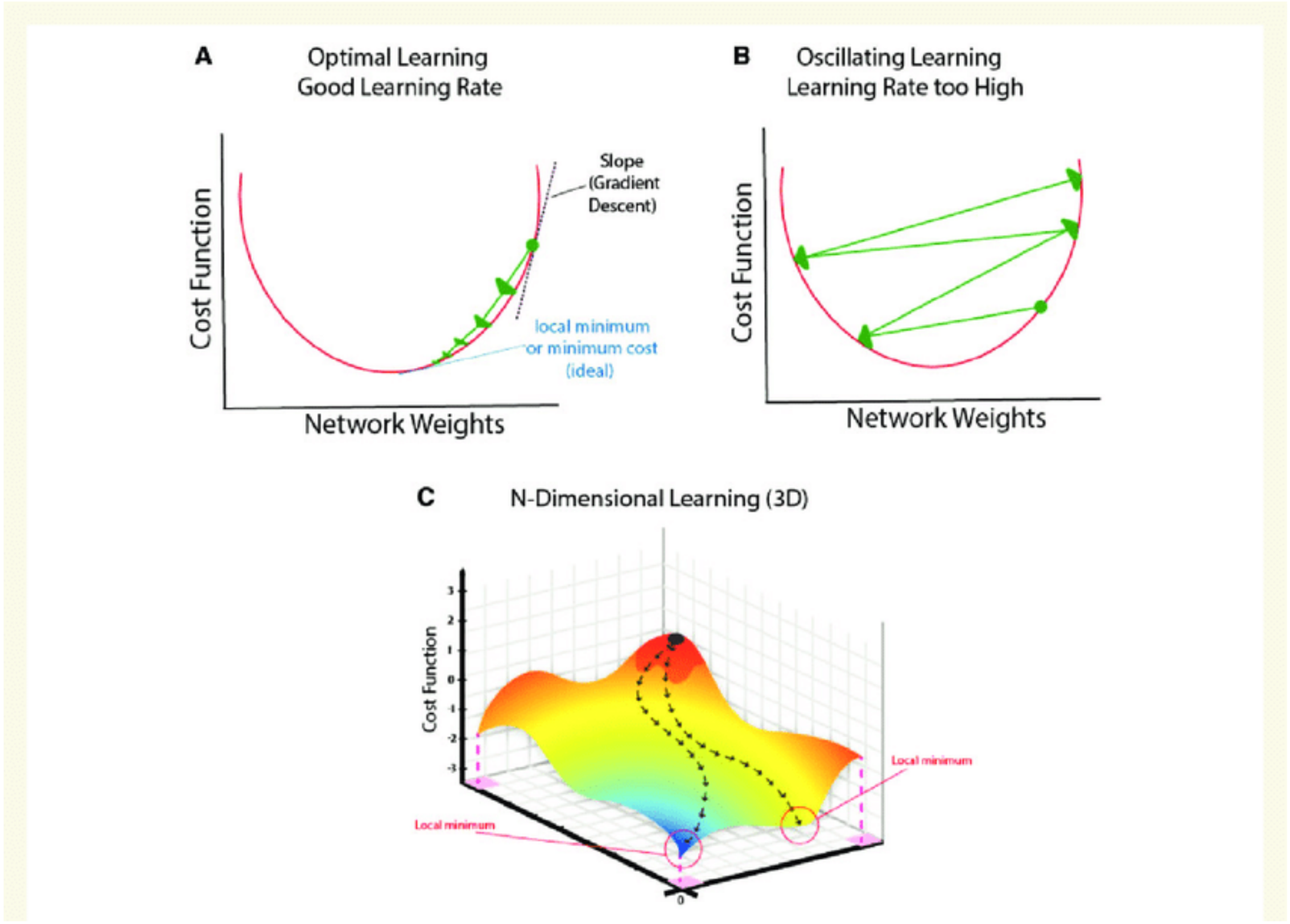
Train your model with a single data point.

Expected behaviour:
Training accuracy = 1
Validation accuracy → Random guess

Otherwise:
DNN too small. Increase its capacity.

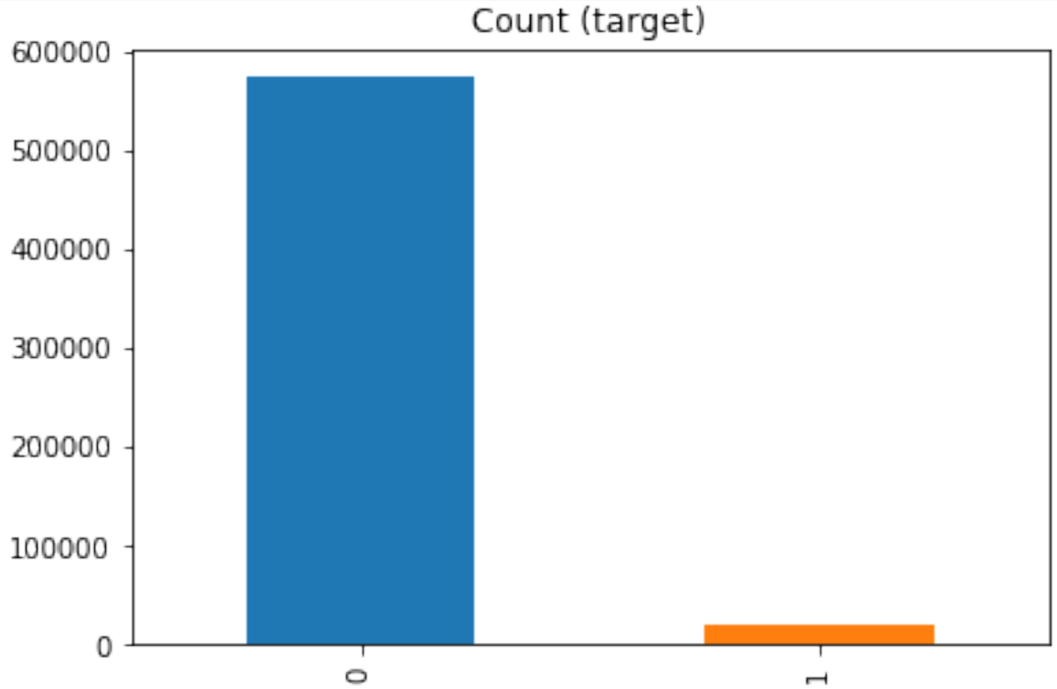


Learning Rates

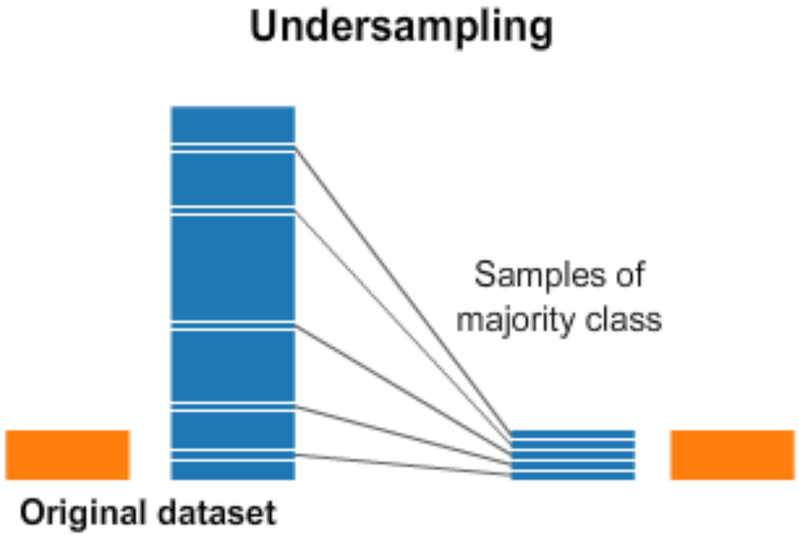


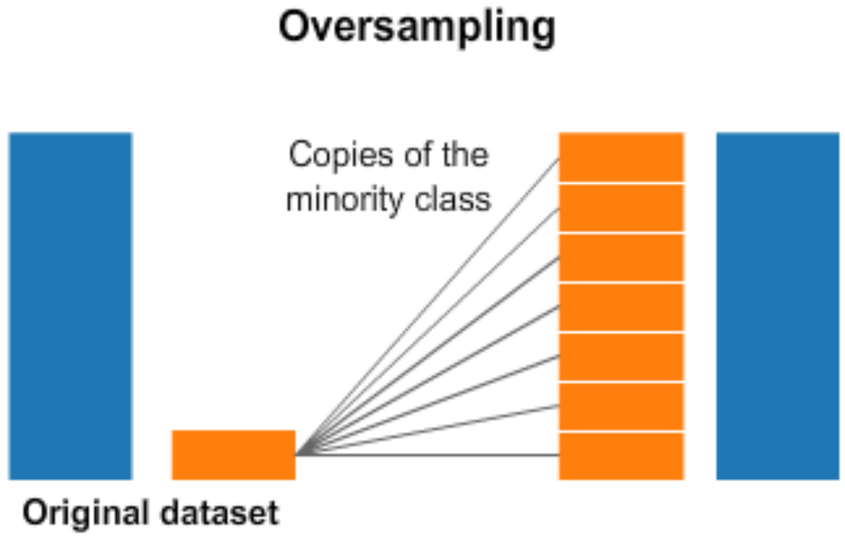
Don’t rely solely on the training/validation loss.

Understand Your Data



In case of **imbalanced datasets**, training/validation losses are not helpful indicators of DNN accuracy.

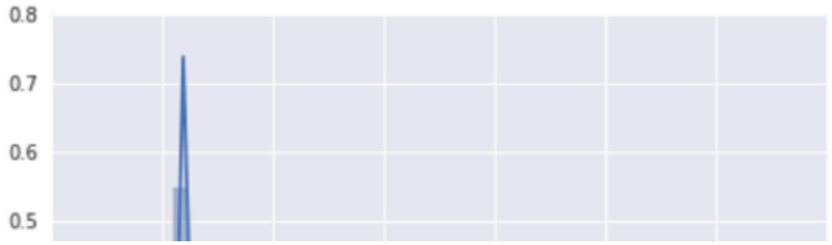


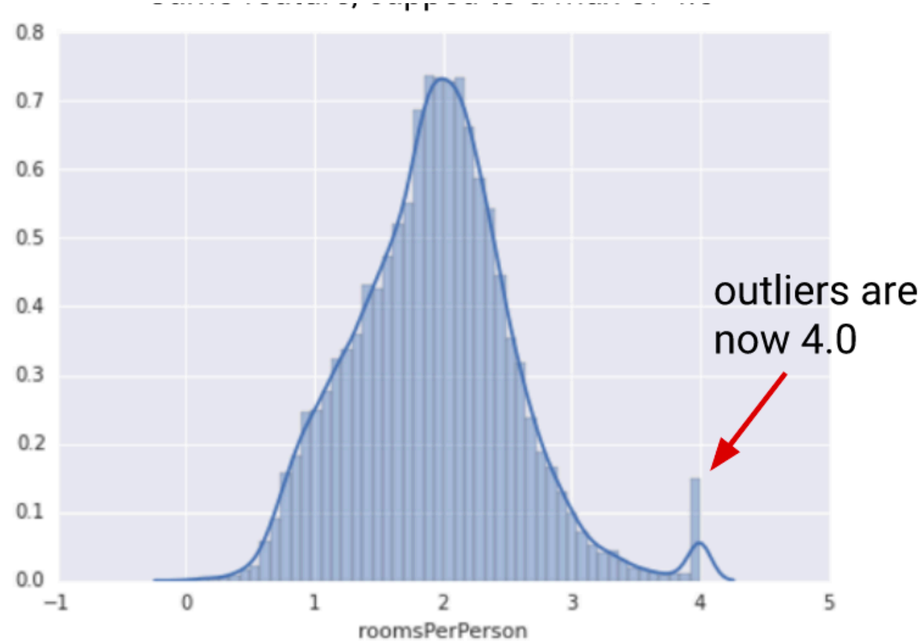


Source: kaggle.com

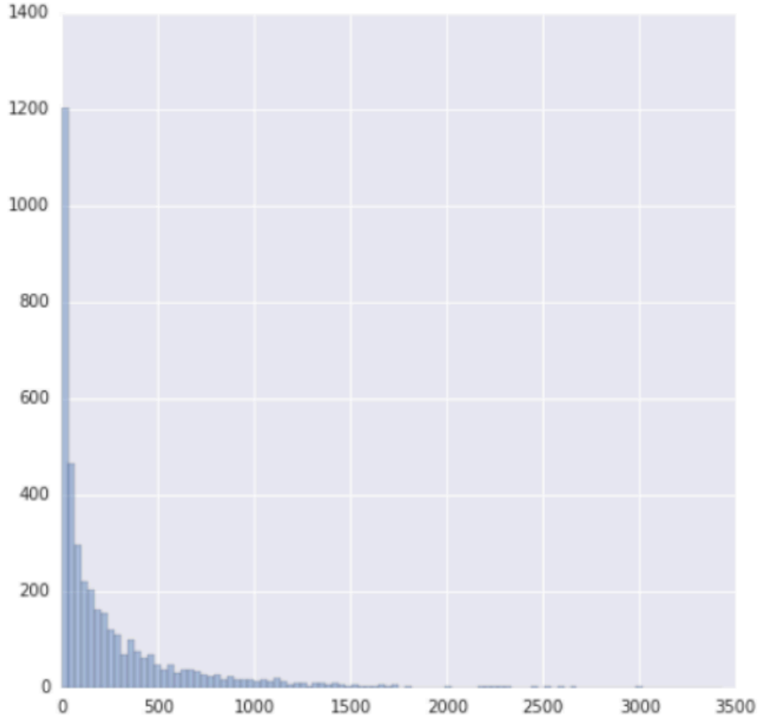
Understand Your Data

In case of **skewed datasets**, training/validation losses are not helpful indicators of DNN accuracy.

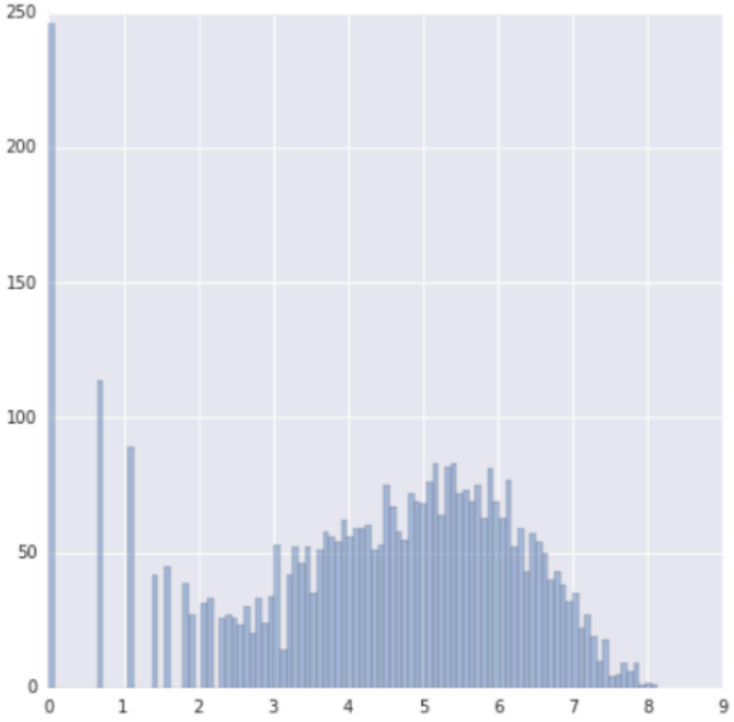




Ratings per movie



Log ratings per movie



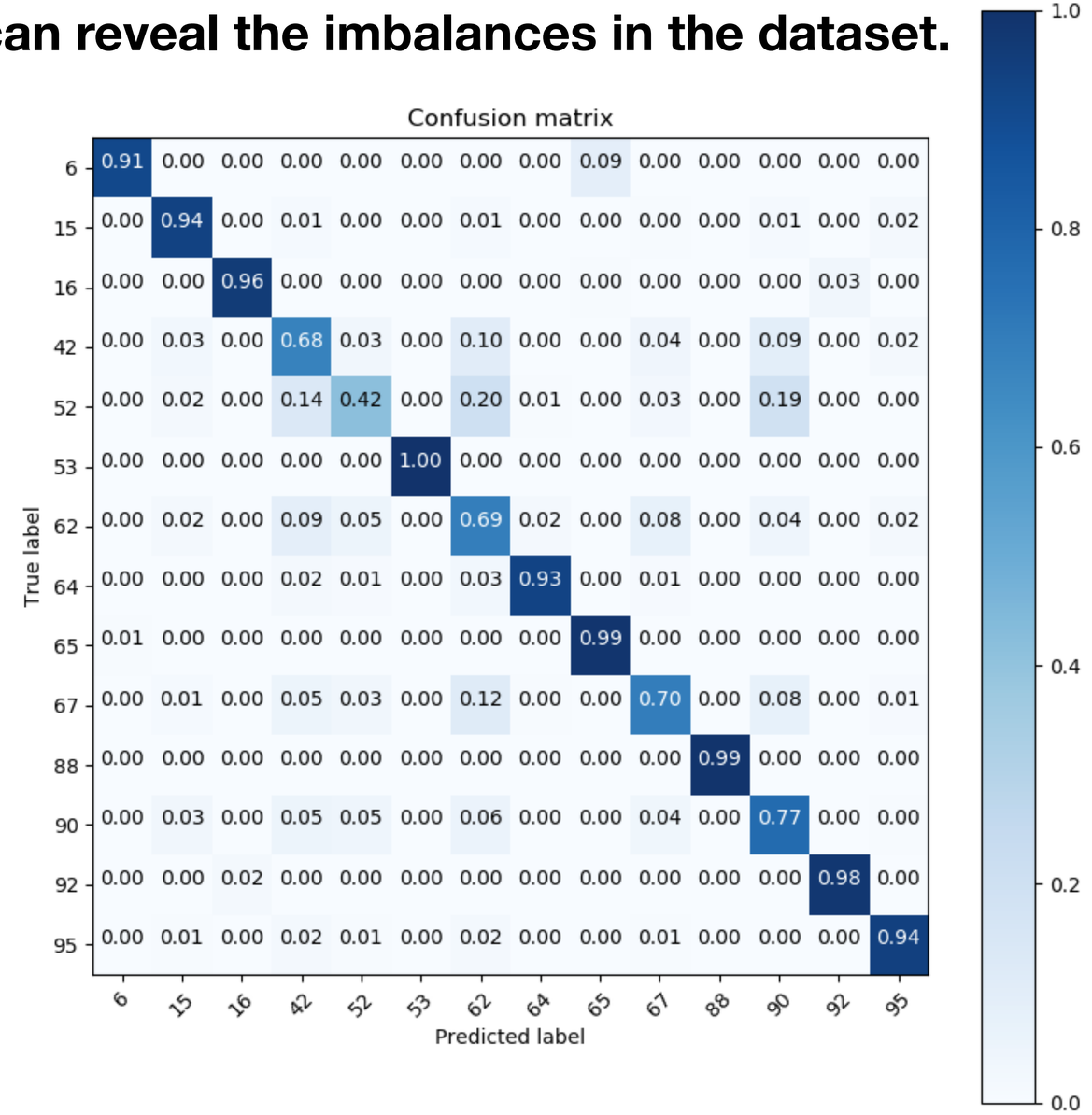
Further reading: sklearn scalers

Source: developers.google.com

Confusion Matrix

A confusion matrix can reveal the imbalances in the dataset.

Prediction outcome	p	n	total
actual value p′	True Positive	False Negative	P′
actual value n′	False Positive	True Negative	N′
total	P	N	



Accuracy = (TP + TN) / (TP + FN + FP + TN)
Precision (p) = TP / (TP + FP)
Recall (r) = TP / (TP + FN)

$$F_1 = \frac{2}{r^{-1} + p^{-1}}$$

Understand Your Activation Function

Use the correct activity function for the correct problem.
E.g., softmax -> multi-class classification
Sigmoid -> binary classification

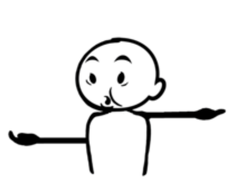
Sigmoid

$$y = \frac{1}{1+e^{-x}}$$

Tanh

$$y = \tanh(x)$$

Step Function



$$y = \begin{cases} 0, & x<n \\ 1, & x \geq n \end{cases}$$

Softplus

$$y = \ln(1+e^{x})$$

ReLU

$$y = \begin{cases} 0, & x<0 \\ x, & x \geq 0 \end{cases}$$

Softsign

$$y = \frac{x}{(1+|x|)}$$

ELU

$$y = \begin{cases} \alpha(e^{x}-1), & x<0 \\ x, & x \geq 0 \end{cases}$$

Log of Sigmoid

$$y = \ln\left(\frac{1}{1+e^{-x}}\right)$$

Swish

$$y = \frac{x}{1+e^{-x}}$$

Sinc

$$y = \frac{\sin(x)}{x}$$

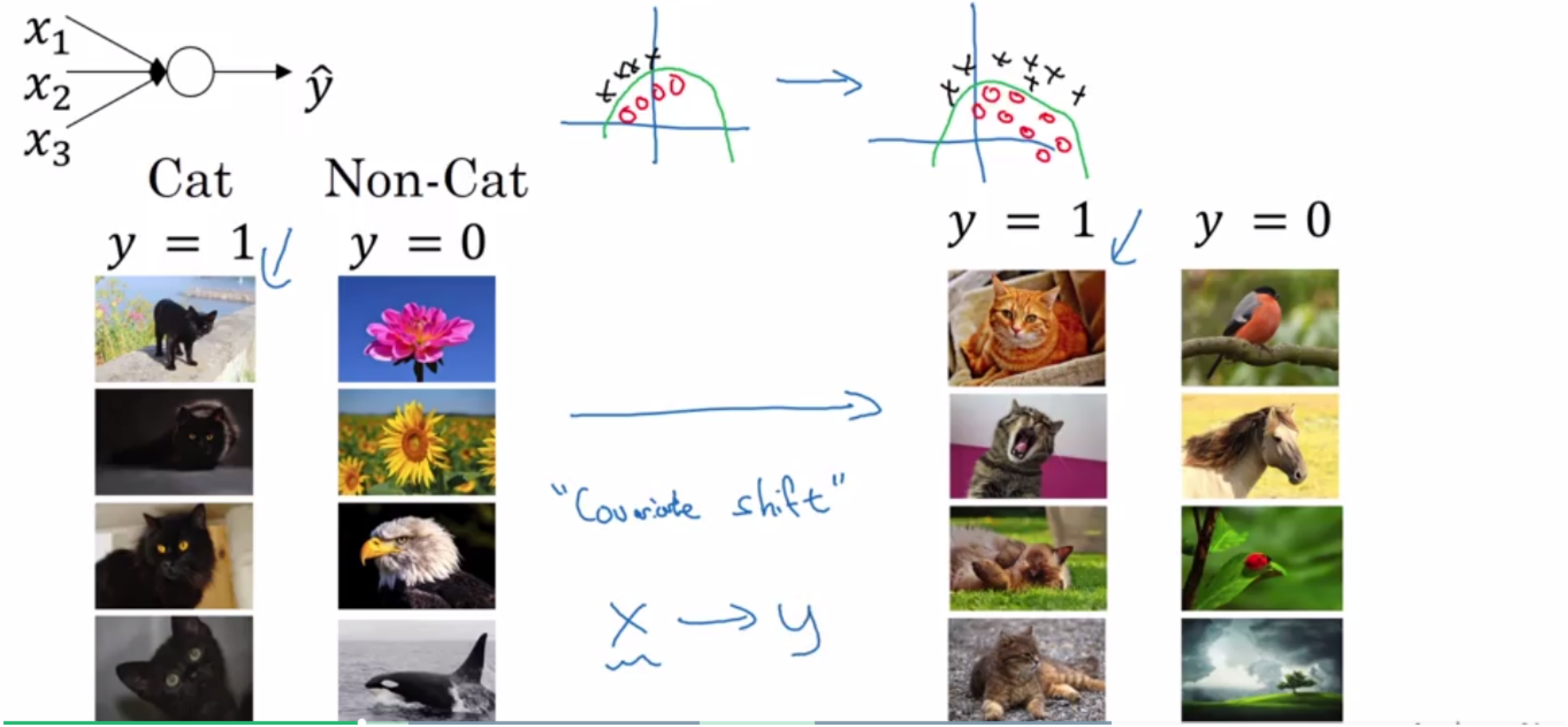
Leaky ReLU

$$y = \max(0.1x, x)$$

Mish

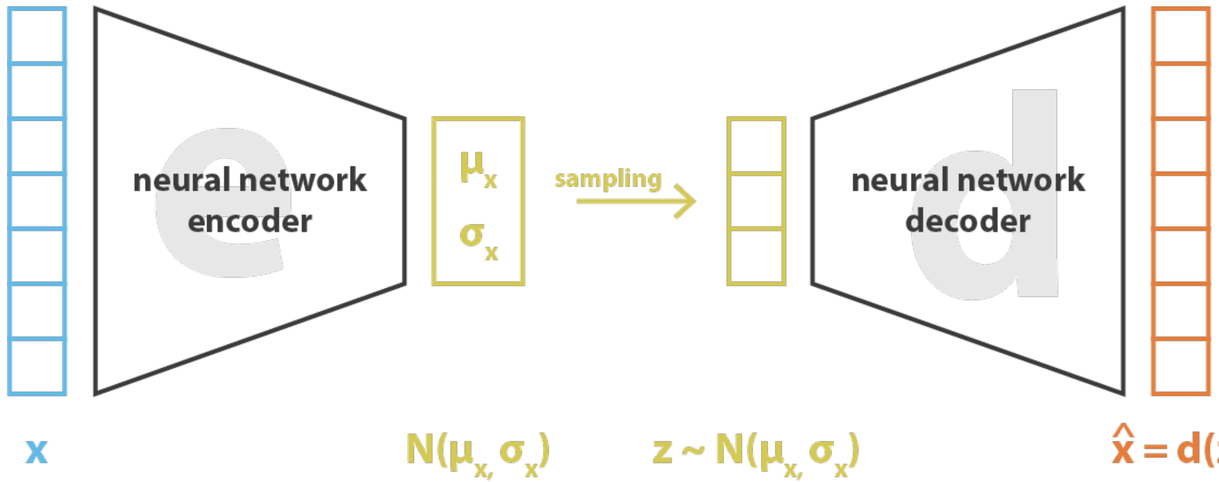
$$y = x(\tanh(\text{softplus}(x)))$$

Batch Size Matters



Understand Your Loss Function

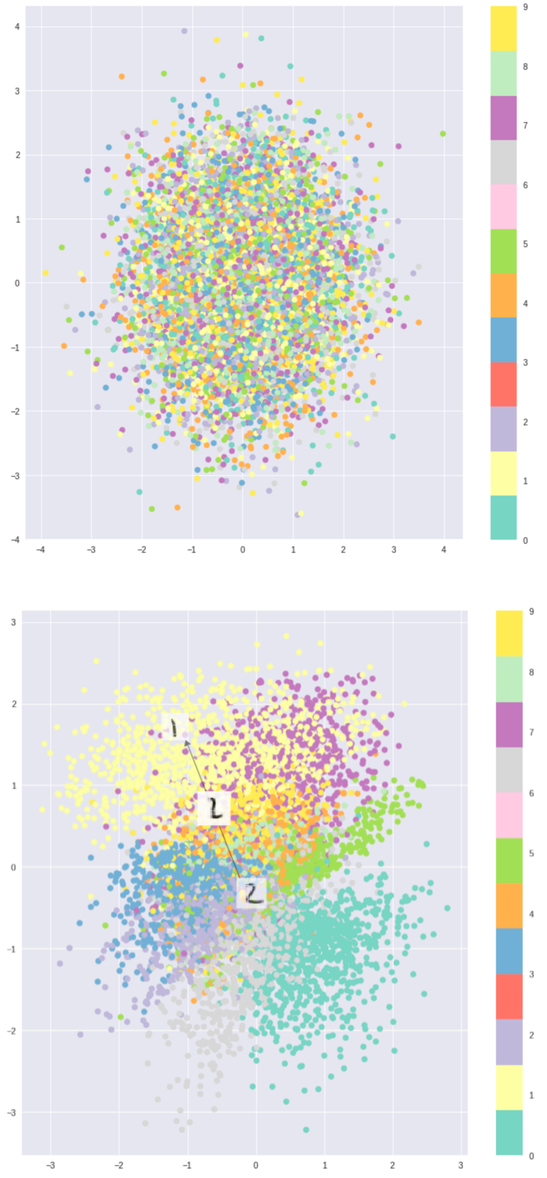
Use an **appropriate** loss function for a given problem.
If **multiple loss functions** are used, make sure that they are of the **same scale**.



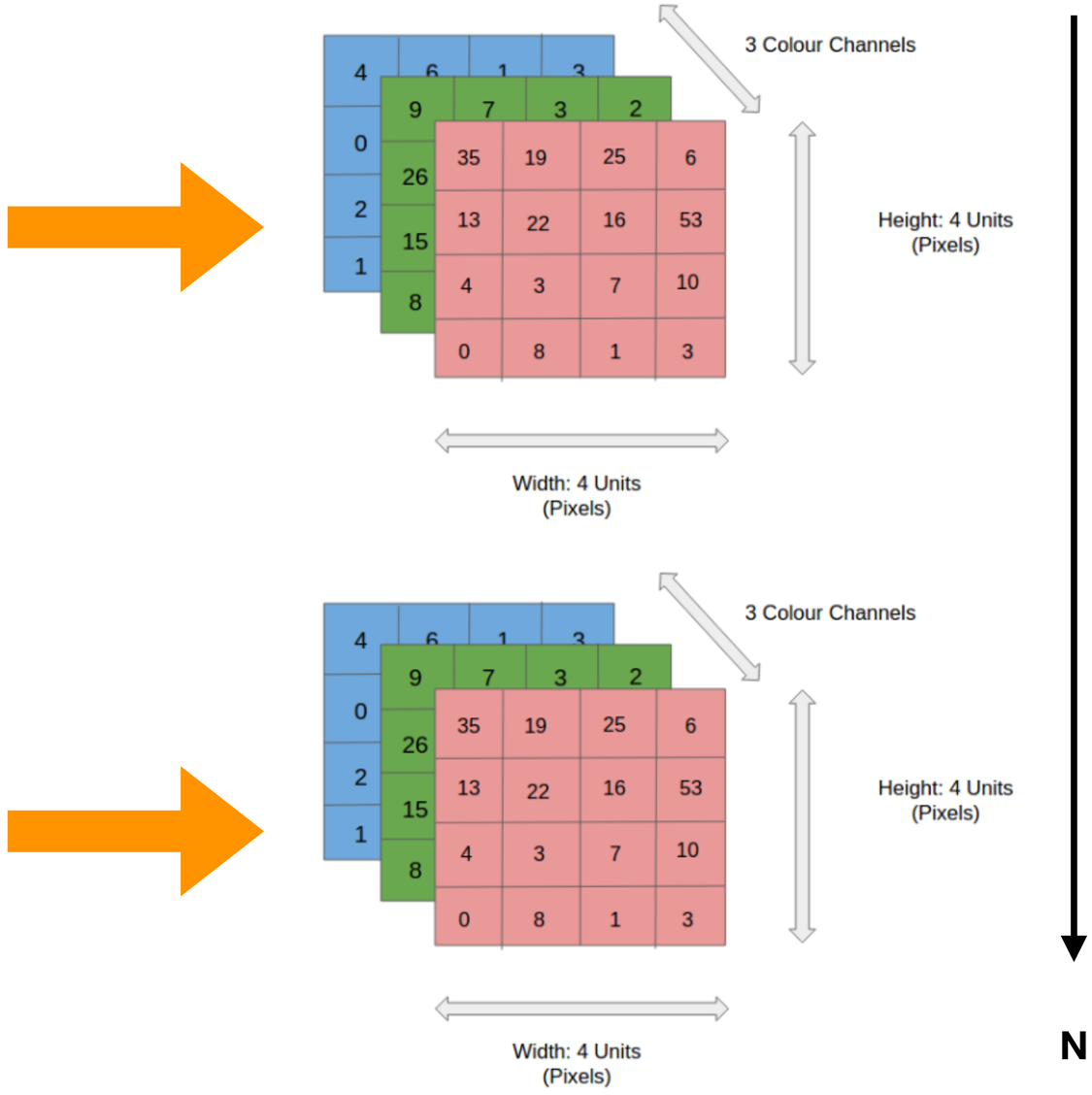
$$\text{loss} = \|x - \hat{x}\|^2 + KL[N(\mu_x, \sigma_x), N(0, I)] = \|x - d(z)\|^2 + KL[N(\mu_x, \sigma_x), N(0, I)]$$

Reconstruction loss

KL loss



Pay Attention to the Data Format

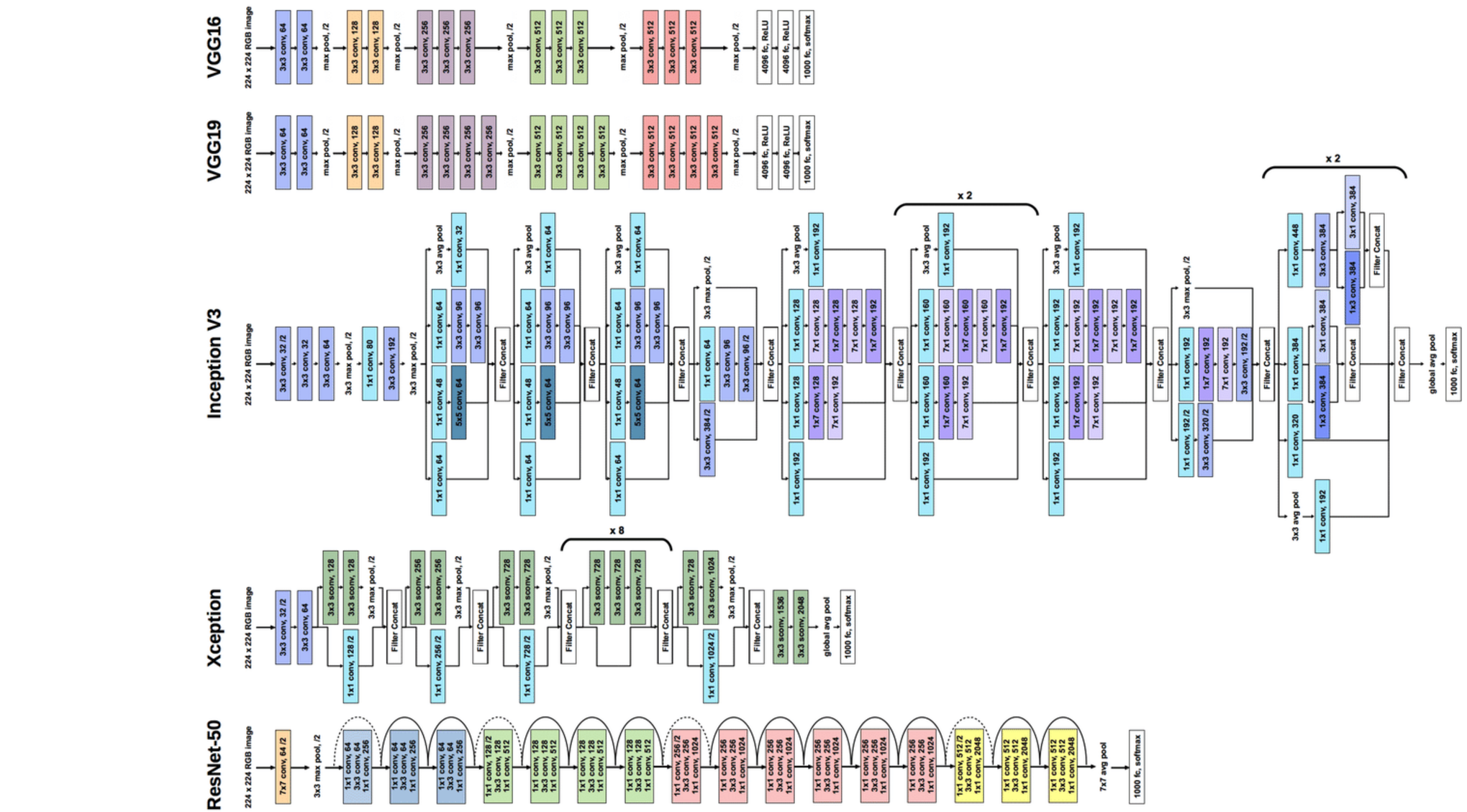


Channel first:
(W, H, C, N)

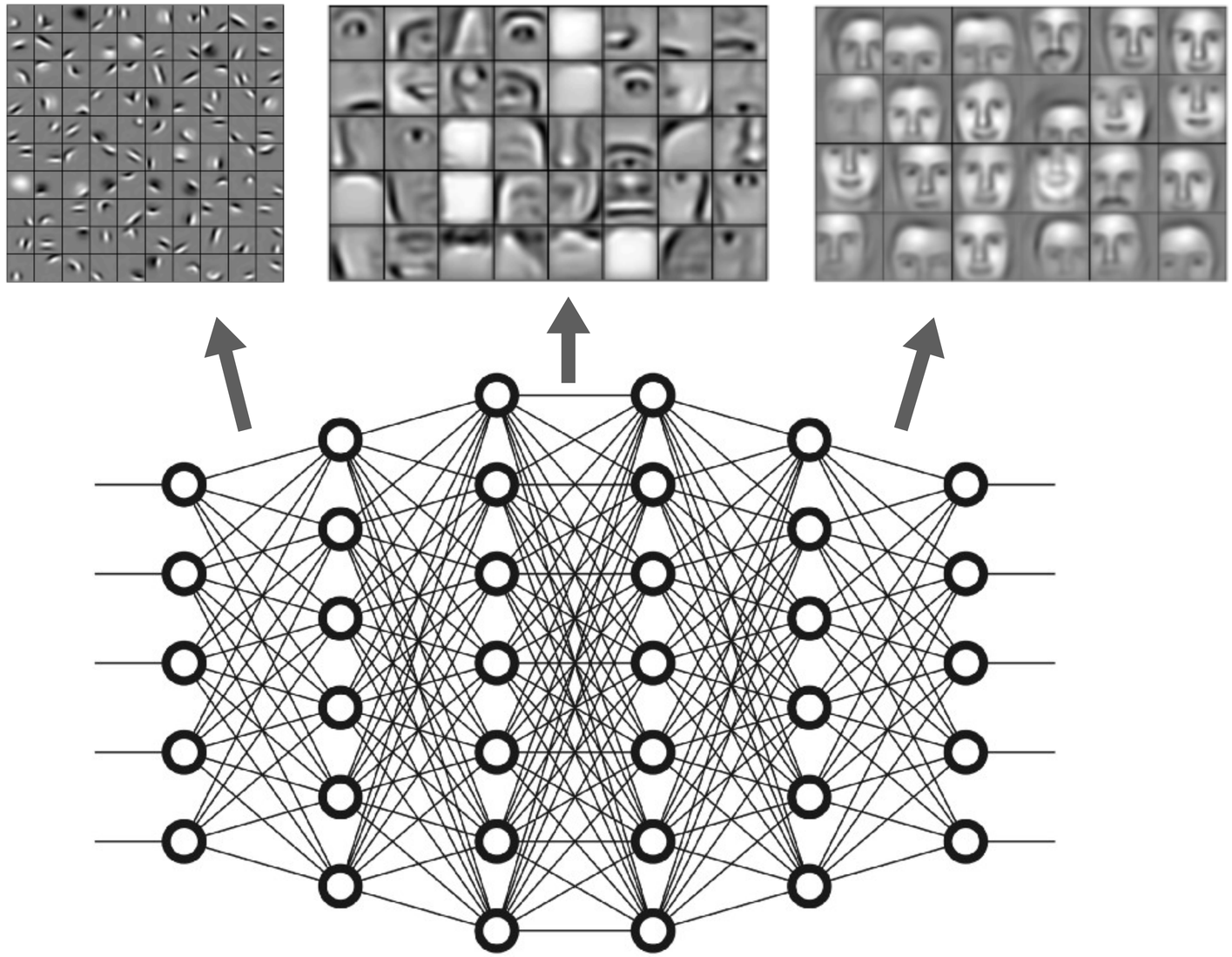
Channel Last:
N, W, H, C

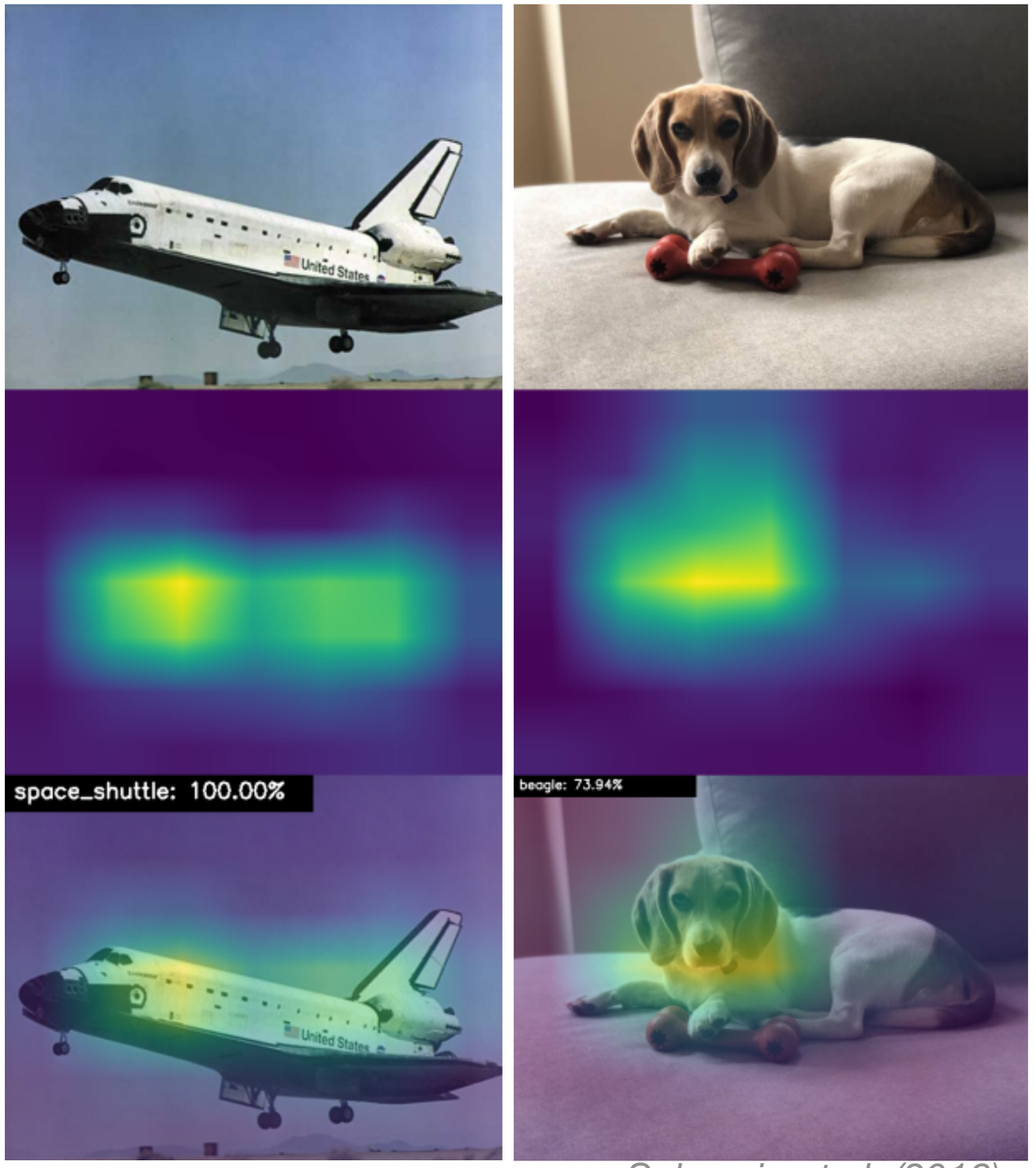
```
model.add(Conv2D(..., data_format='channels_first'))
```

Compare with baseline models



Inspect Intermediate Outputs





Selvaraju et al. (2016)