Deep Learning

Lecture 4: Training Models - Challenges and Applications

Chris (Shuang) Chen

shuang.chen@durham.ac.uk

Durham University

Durham University

Lecture Overview

Convolutional Architectures and Backbones

Building Blocks of Neural Networks

Improved Training

Challenges
- Failures of learning
- Bias in deep learning
- Combatting algorithmic bias

Adversarial examples
- Definition
- Defence

Interpreting neural networks
- Feature visualisation
- Explainability

ImageNet

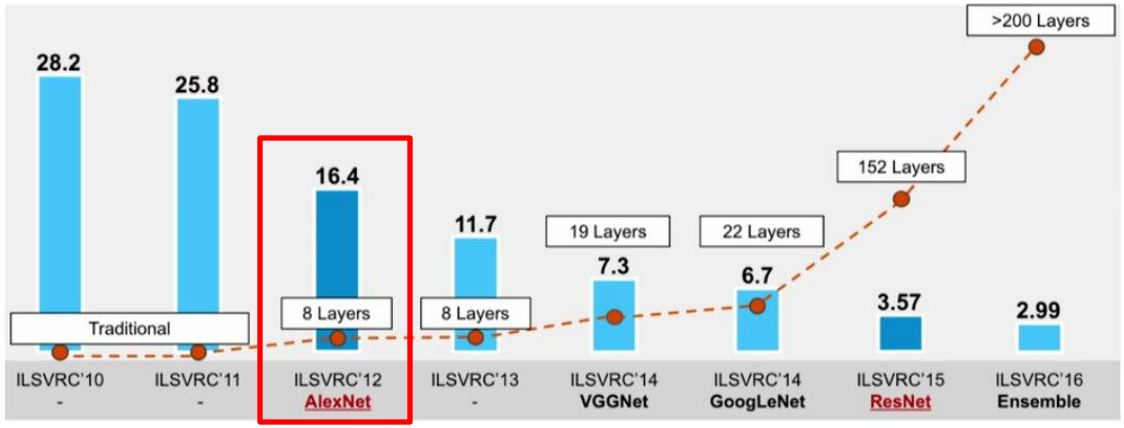
Deep Models and Big Data:

- ImageNet Large Scale Visual Recognition Challenge.
  - More than 14 million images for different tasks.

http://www.image-net.org/

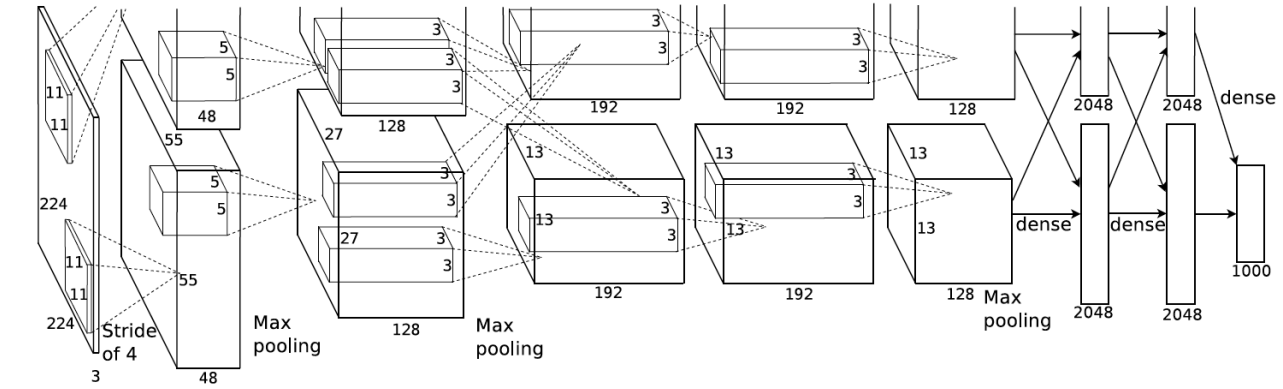
Architecture Design

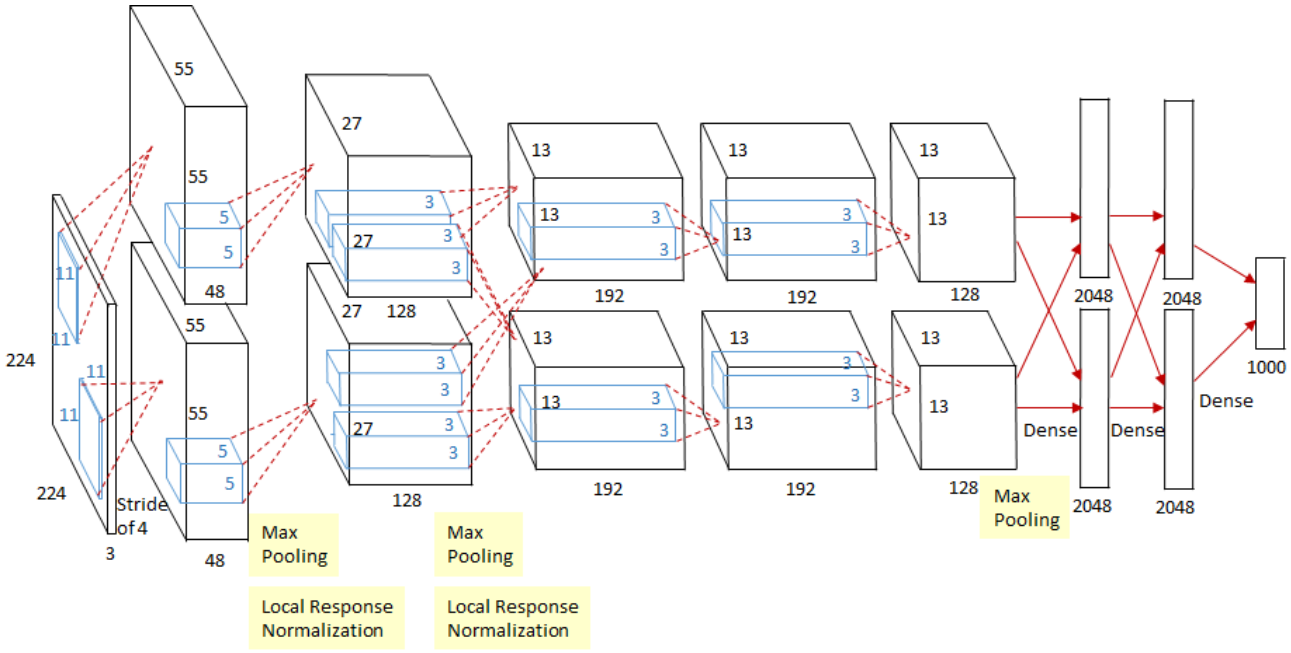
AlexNet [Krizhevsky et. al, 2012]



First CNN-based winner of the ImageNet Large Scale Visual Recognition Challenge in 2012.

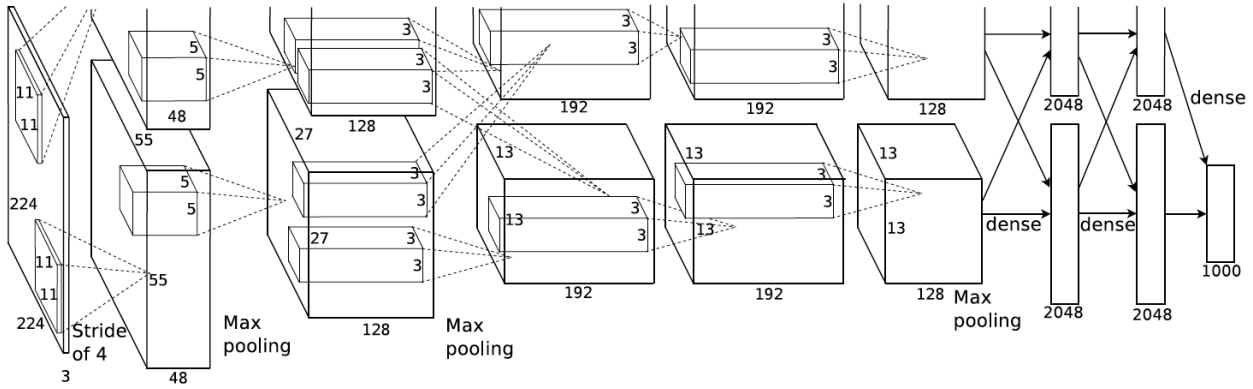
- Trained on GTX 580 with 3GB of memory.
- Network and feature maps are spread across two GPUs.
- Cross-talk across certain layers.





Architecture Design

AlexNet [Krizhevsky et. al, 2012]



Input $224 \times 224 \times 3$

1. CONV: K=96×11×11, S=4, P=0
2. MAXPOOL: K=3×3, S=2
3. NORM: Normalisation Layer (No longer used)
4. CONV: K=256× 5×5, S=1, P=2
5. MAXPOOL: K=3×3, S=2
6. NORM: Normalisation Layer (No longer used)
7. CONV: K=384× 3×3, S=1, P=1
8. CONV: K=384× 3×3, S=1, P=1
9. CONV: K=256× 3×3, S=1, P=1
10. MAXPOOL: K=3×3, S=2
11. FC: 4096D
12. FC: 4096D
13. FC: 1000 (number of classes in ImageNet)

Local Response Normalisation Layers:

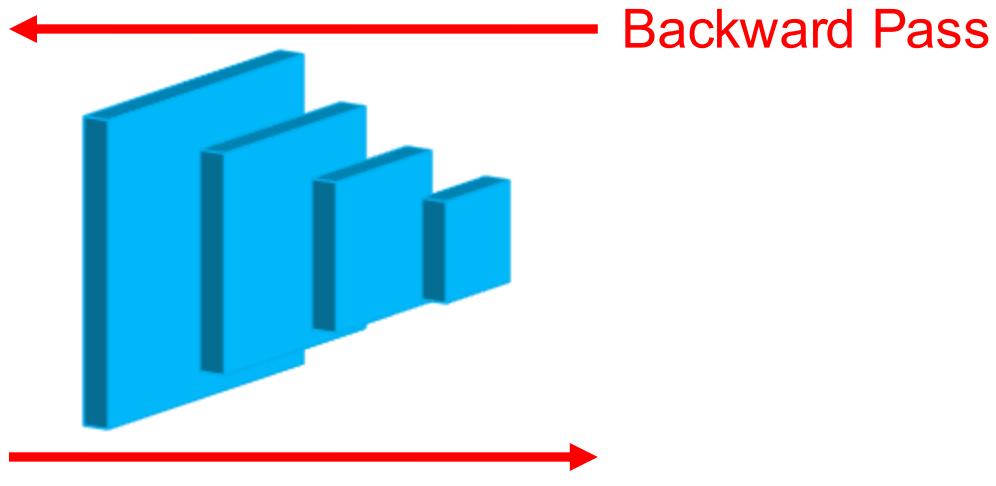
Normalising the response of the ReLU across neighbouring channels.
(Demonstrated to be ineffective)

Batch Normalisation is common.

Building Blocks

Batch Normalisation

All layers are updated in the backward pass.
Every layer assumes other layers are fixed.



The weights of a layer can be updated based on the expectation that the prior layer produces values with a given distribution. However, this distribution is likely to change once that layer is updated.

Problem: Internal Covariate Shift

(change in the distribution of input during training)

Solution: Batch Normalisation

(scaling the output of every layer to a standard distribution)

Building Blocks

Batch Normalisation

Layer outputs are rescaled such that they have a mean of *zero* and a standard deviation of *one*, *i.e.* a **standard Gaussian**.

$$\hat{x}^{(k)} = \frac{x^{(k)} - \mathrm{E}[x^{(k)}]}{\sqrt{\mathrm{Var}[x^{(k)}]}}$$

The layer is then allowed to learn two new parameters (γ and β), using which identity mapping can be recovered if the learning process needs it.

$$\hat{x}^{(k)} = \gamma^{(k)}\, \hat{x}^{(k)} + \beta^{(k)}$$

At test time, a running mean and variance calculated during training is used for normalisation.

Building Blocks

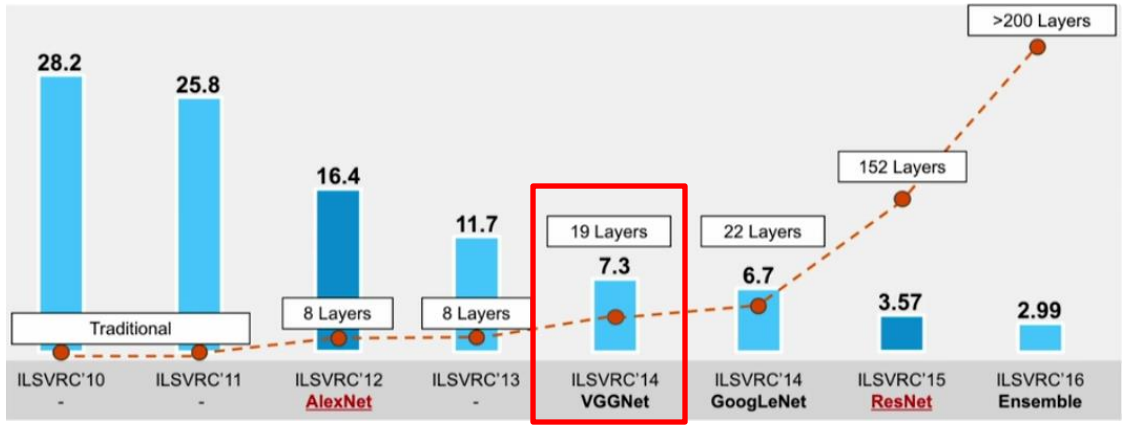
Types of Normalisation

There are various other normalisation methods (for further reading):

- Weight Norm: separates the norm of the weight vector from its direction.
- Layer Norm: normalises the inputs across the features (instead of batches).
- Instance Norm: normalises the inputs across each channel.
- Group Norm: normalises over group of channels for each training examples.
- Batch-Instance Normalisation
- Switchable Normalisation
- … .

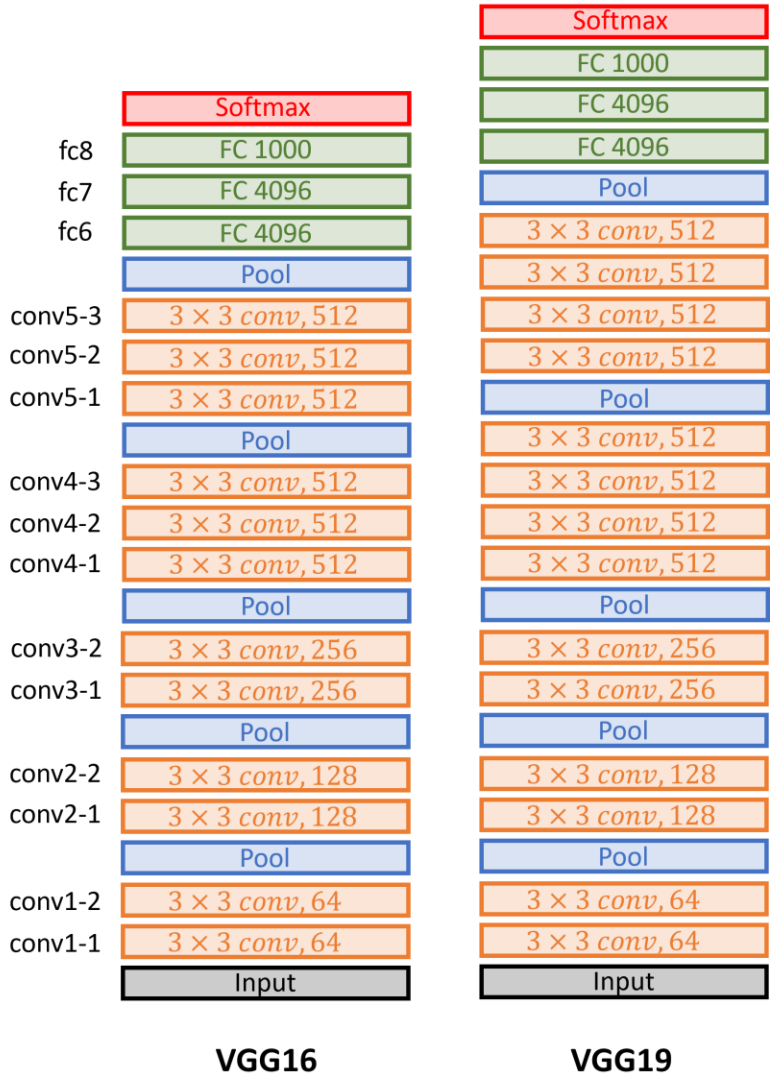
Architecture Design

VGG [Simonyan and Zisserman, 2014]



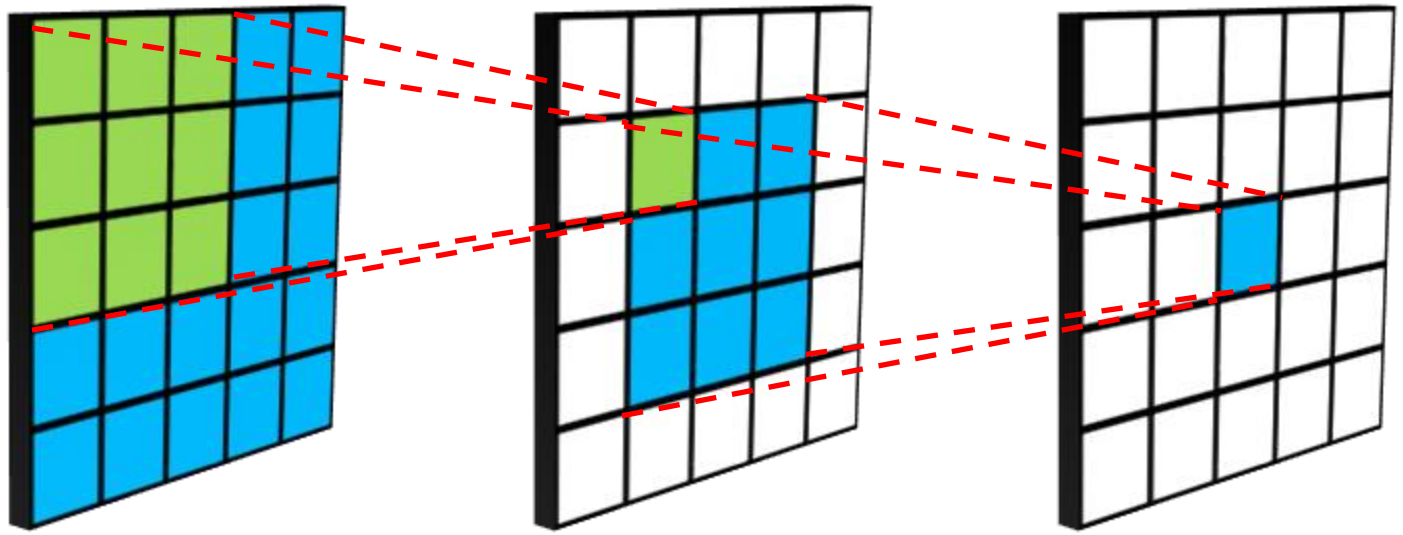
Second place in the ImageNet Large Scale Visual Recognition Challenge in 2014.

- Deeper: more layers (16 and 19)
- Smaller kernels: stacks of 3×3 convolutions
  - Fewer parameters
  - Same effective receptive field



Receptive Field

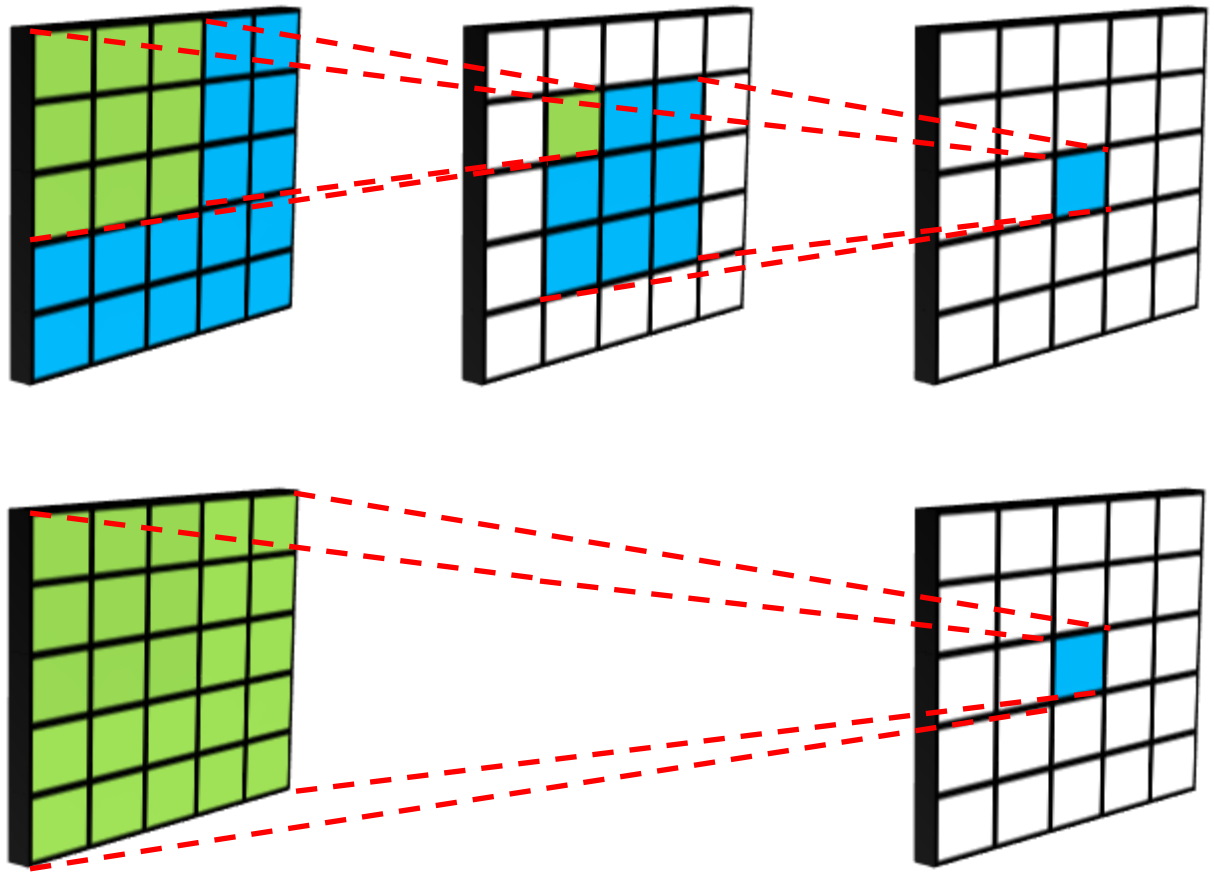
- **Biology**: portion of sensory space that can elicit neuronal responses, when stimulated.
- **Deep learning**: size of the region in the input that produces the feature at any given layer.



Decision-making neurons should be able to see the entire input.

Receptive Field

- Two 3x3 convolutional layers VS One 5x5 convolutional layer

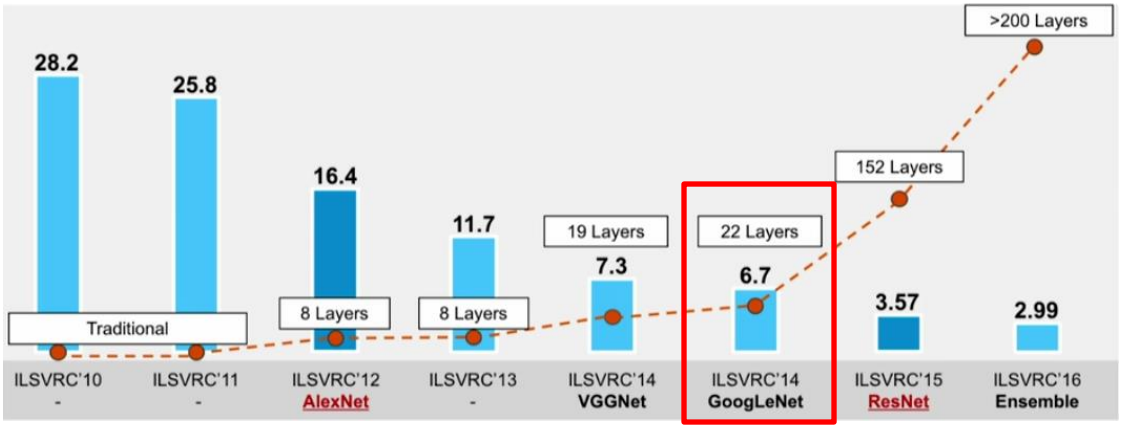


Think

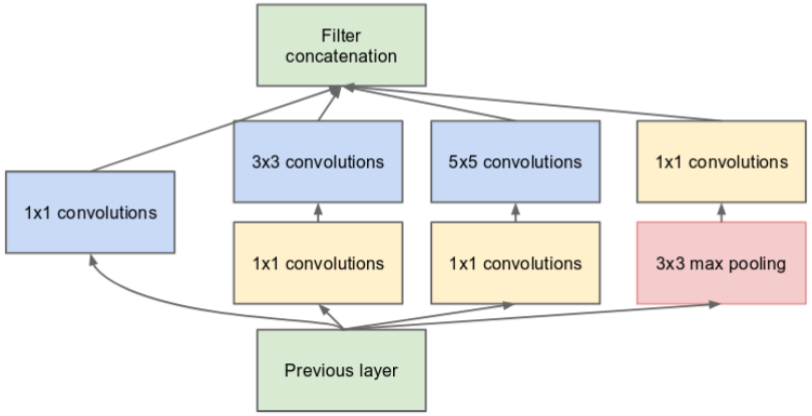
Differences?

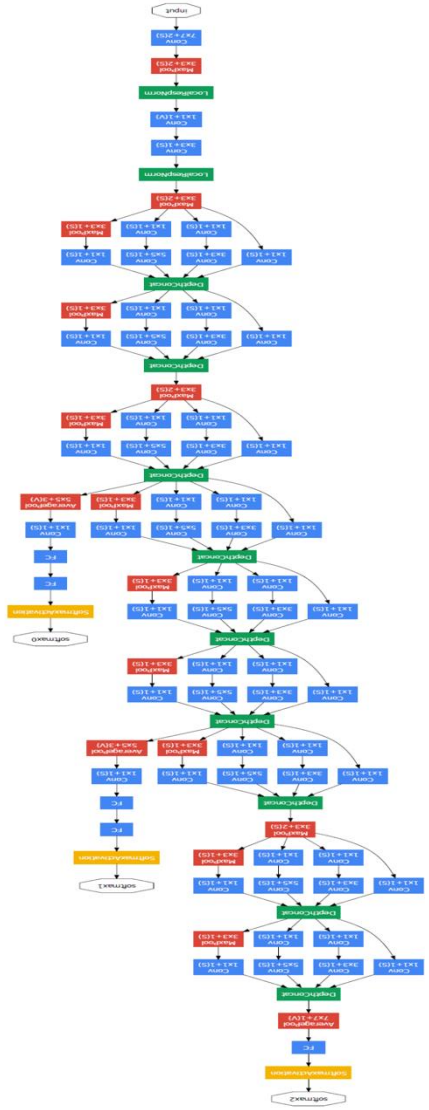
Architecture Design

GoogLeNet (Inception) [Szegedy et al., 2014]



Winner of the ImageNet Large Scale Visual Recognition Challenge in 2014.



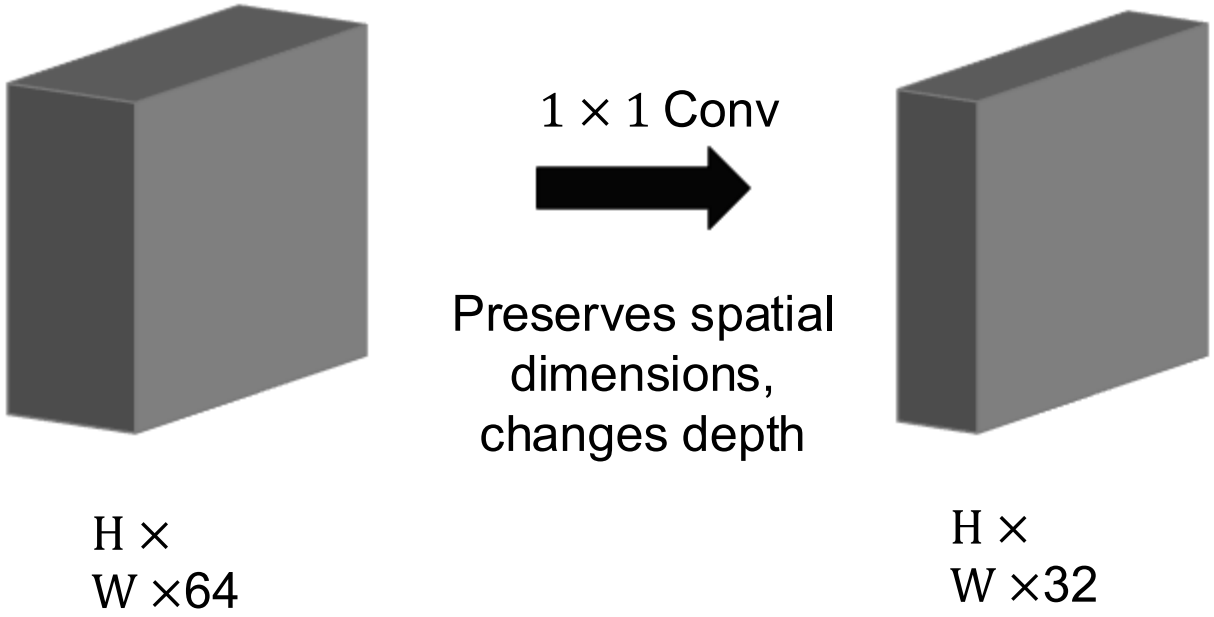


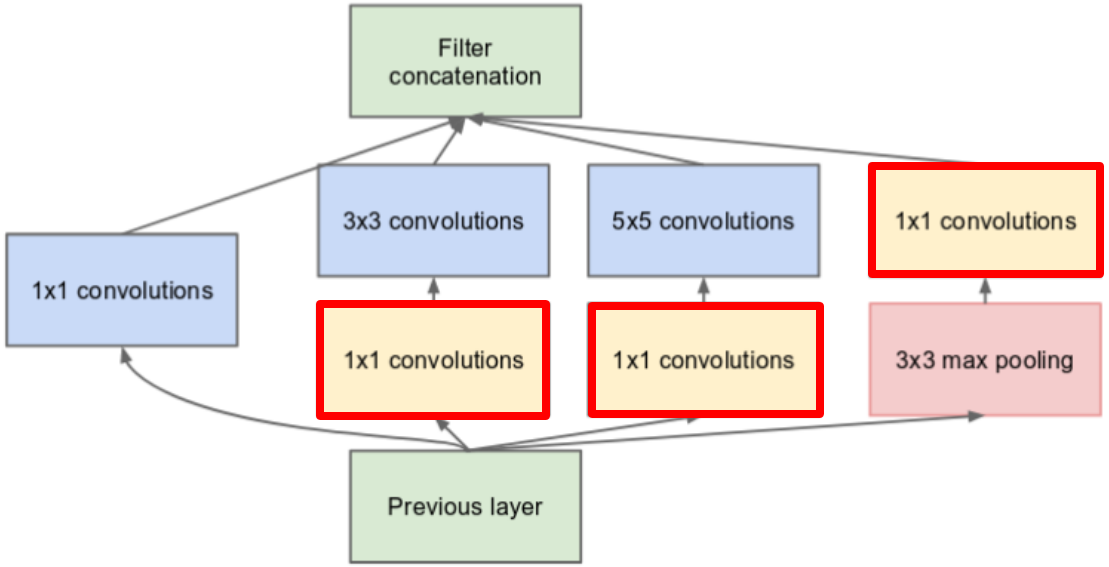
- Deeper but with fewer parameters
- No Fully-connected layers
- Uses “Inception” modules
  - Multiple kernel operations in parallel

Building Blocks

Dimensionality Reduction

- Different convolution operations in parallel enable capturing localities and structures of different sizes in the image.
- 1×1 Convolutions can reduce feature dimensionality to better control the depth of features before they are concatenated in the output.



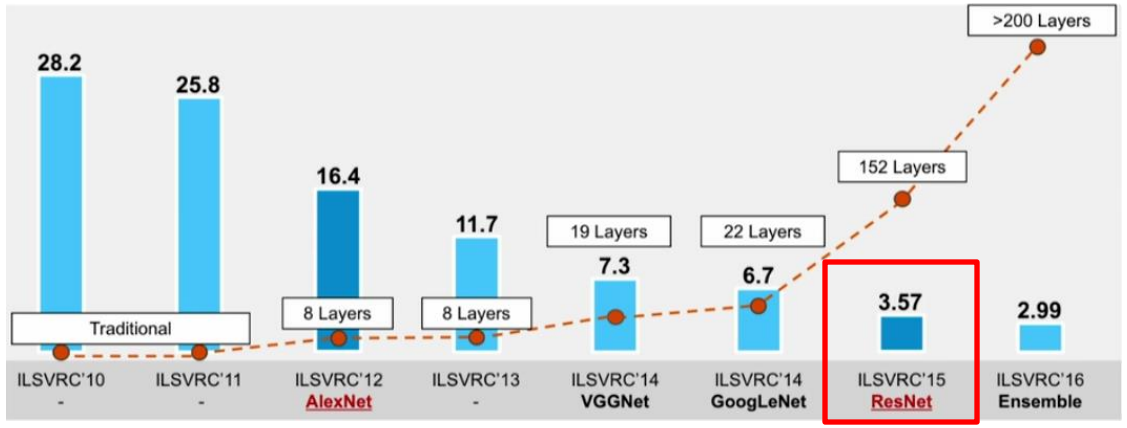


Architecture Design

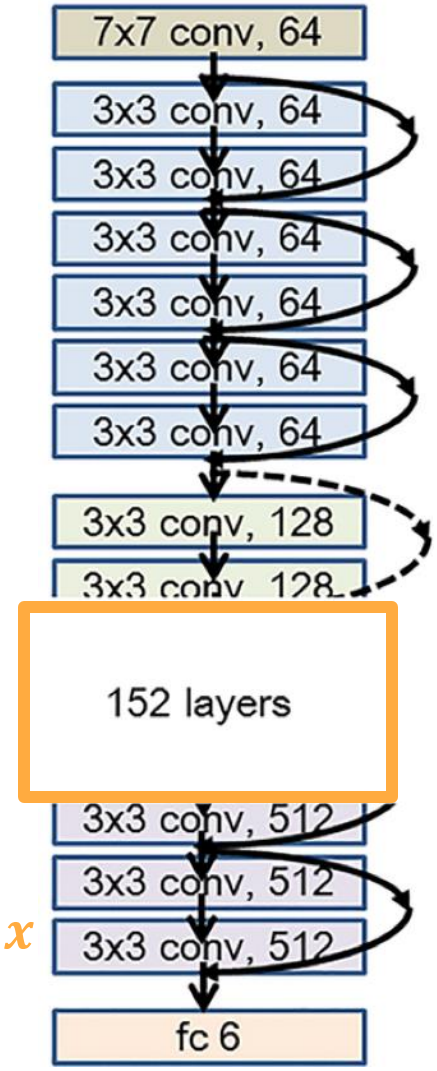
ResNet [He et al., 2015]

Very deep.

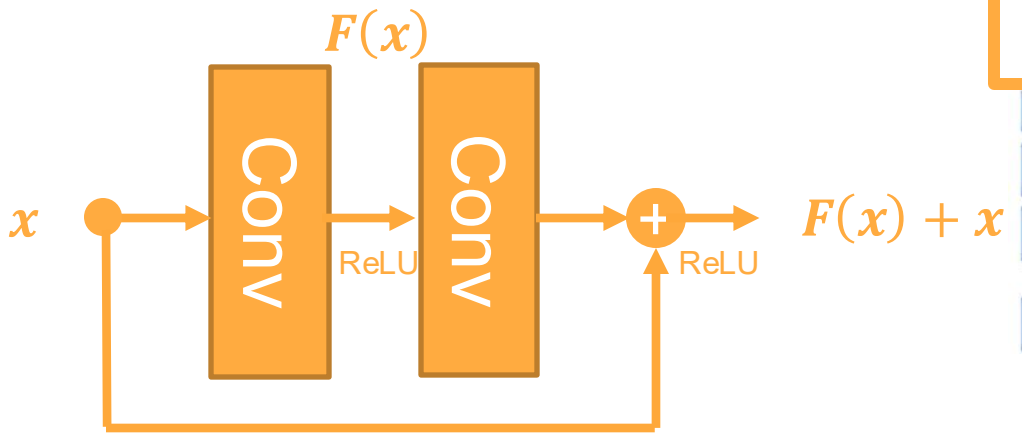
- Up to 152 layers.
- Optimisation is difficult.



Winner of the ImageNet Large Scale Visual Recognition Challenge in 2015.



- Residual modules are introduced to learn a residual mapping instead of a direct mapping.

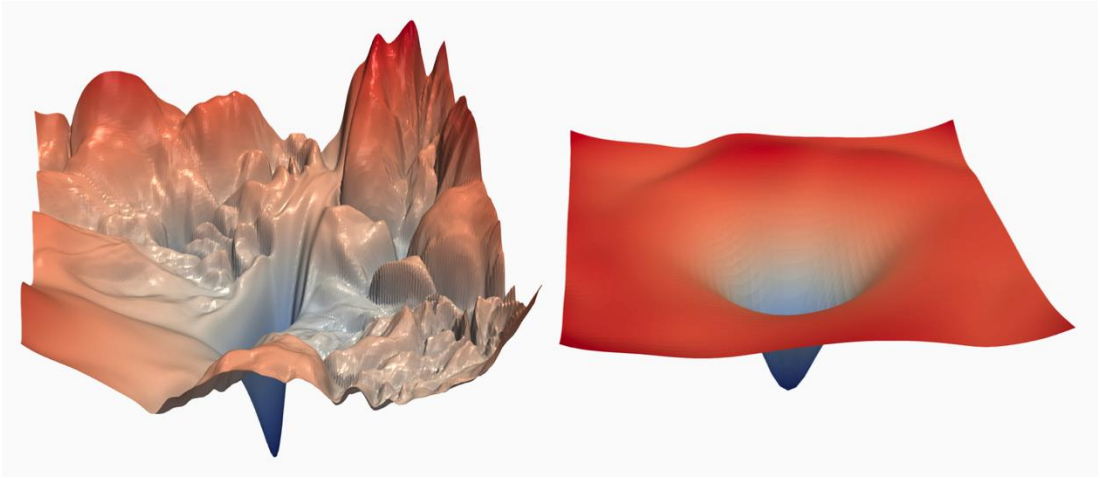


- No fully-connected layers.
- Depths of 18, 34, 50, 101, 152.

Building Blocks

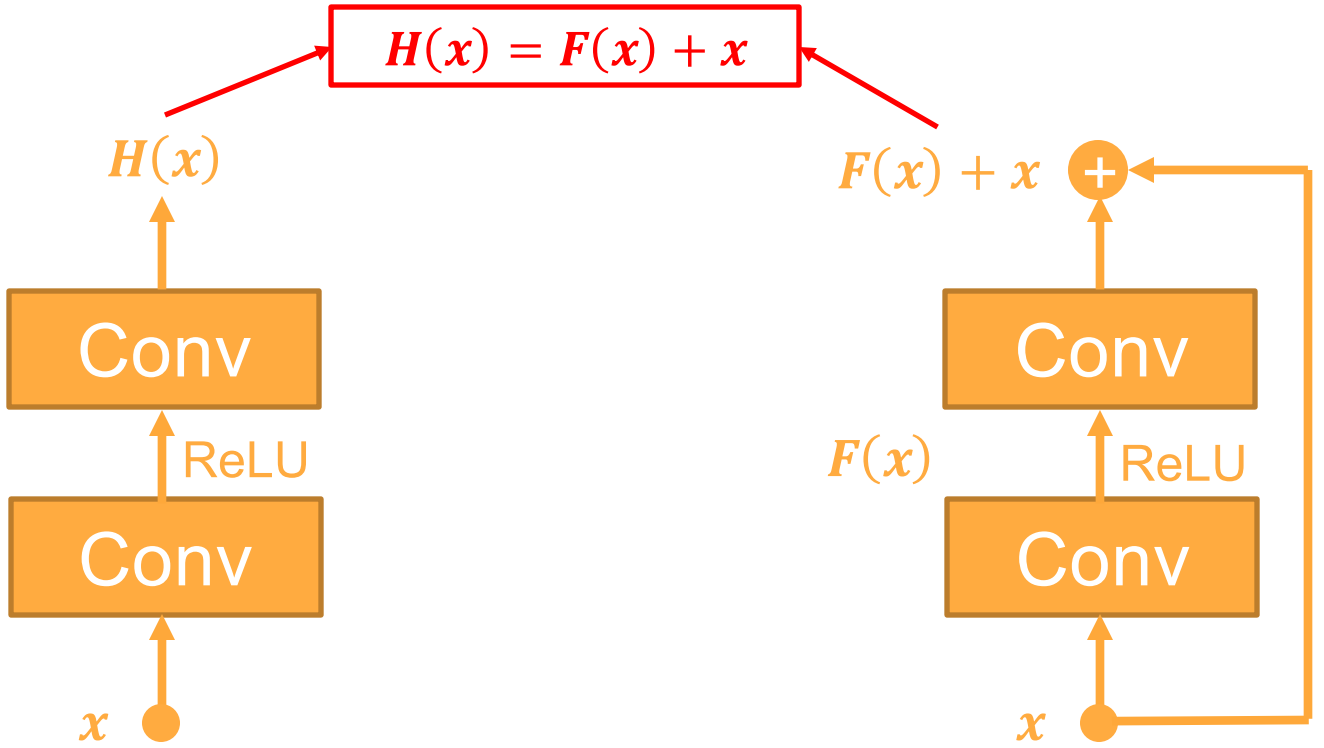
Residual Connections

- A “plain layer” learns a direct mapping from the input to the output.
- A **Residual block** learns a residual to the identity mapping.



Loss landscape without residuals

Loss landscape with residuals

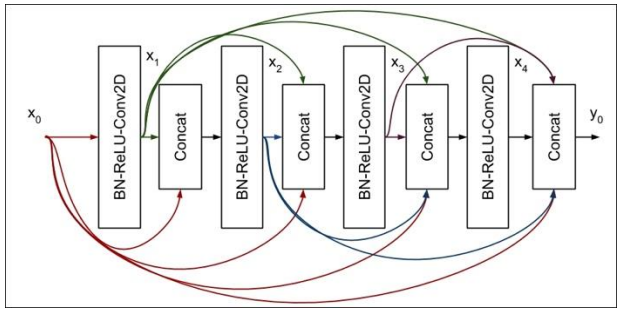


Plain Layer

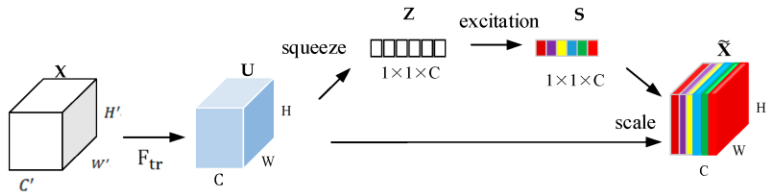
Residual Layer

Architecture Design The list is long…!

- DenseNet

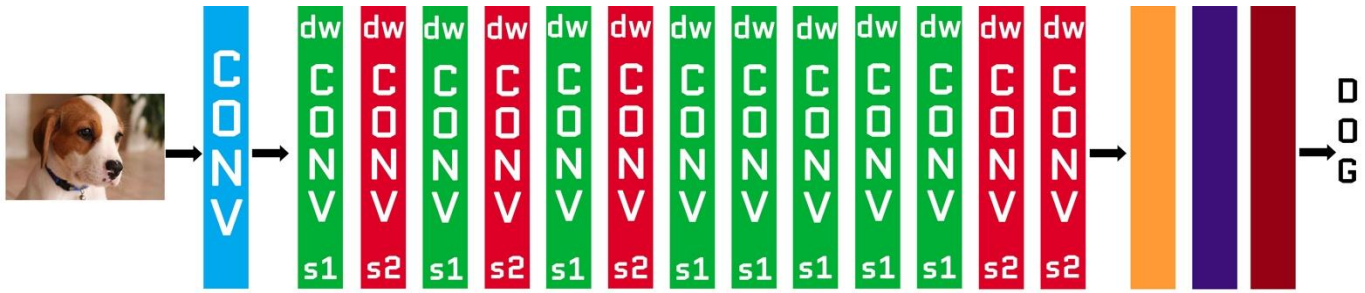


- SENet



- MobileNet

Depthwise Separable Convolutions



- ConvNeXt
- MaxVit
- MNASNet
- RegNet
- ResNeXt
- ShuffleNet
- SqueezeNet
- SwinTransformer
- VisionTransformer
- Wide ResNet
- ….

https://pytorch.org/vision/stable/models.html

Better Training

Poor Model Performance

Poor Performance is often caused by:

- Wrong architecture design
- Model Capacity
  - how complicated a pattern or relationship can the model express?
  - how complex a function can the model approximate?
- Wrong hyperparameters
- Overfitting
  - when the model fits a particular set of data too closely and does *not* generalise to future unseen data.
  - Model is too complex OR dataset is too small.
    - Model performs well on the training data.
    - Model cannot perform on the test data it has not seen yet.
    - *essentially memorising noise patterns in the training data.*

Better Training

Poor Model Performance

Overfitting (an example):

Find the next number in the sequence

1, 3, 5, 7, ?

We can use a quartic function to solve this!

$$f(x) = \frac{18111}{2}x^4 - 90555x^3 + \frac{633885}{2}x^2 - 452773x + 217331$$

$x =$ Index in sequence

$f(1) = 1$ $\quad$ $f(3) = 5$

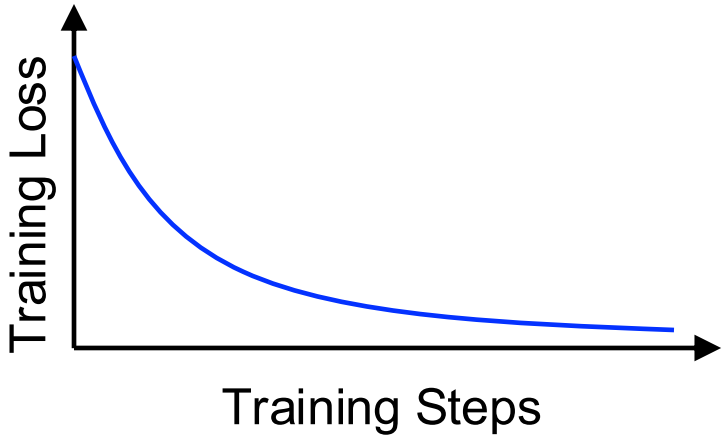
$f(2) = 3$ $\quad$ $f(4) = 7$

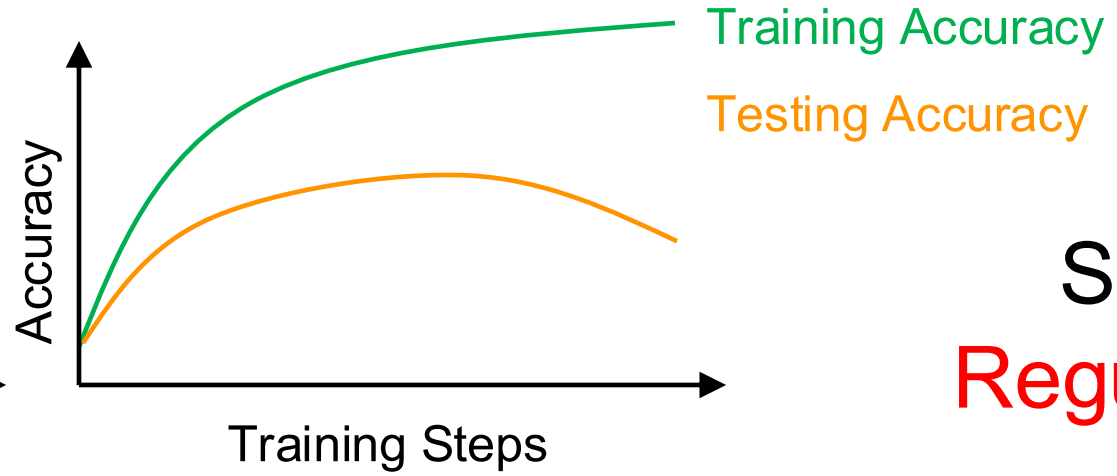
$f(5) = 217341$

Better Training

Poor Model Performance

Overfitting is very common in deep learning applications.





Solution:
Regularisation

L_2 Regularisation

$$R(W) = \sum_i \sum_j W_{ij}^2$$

L_1 Regularisation

$$R(W) = \sum_i \sum_j |W_{ij}|$$

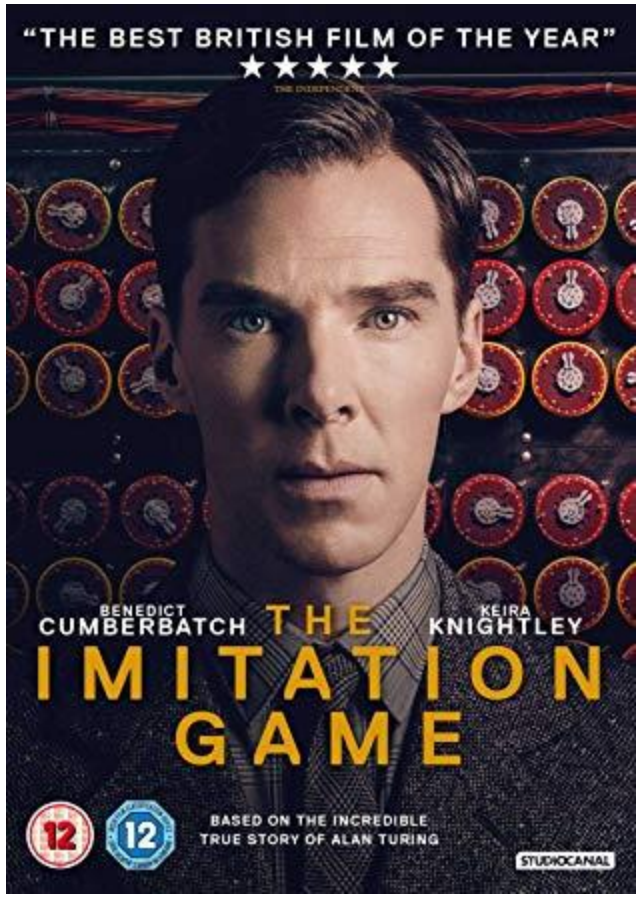
Elastic net

$L_1 + L_2$

Better Training

Poor Model Performance

Overfitting somehow could be useful in specific application...



Better Training

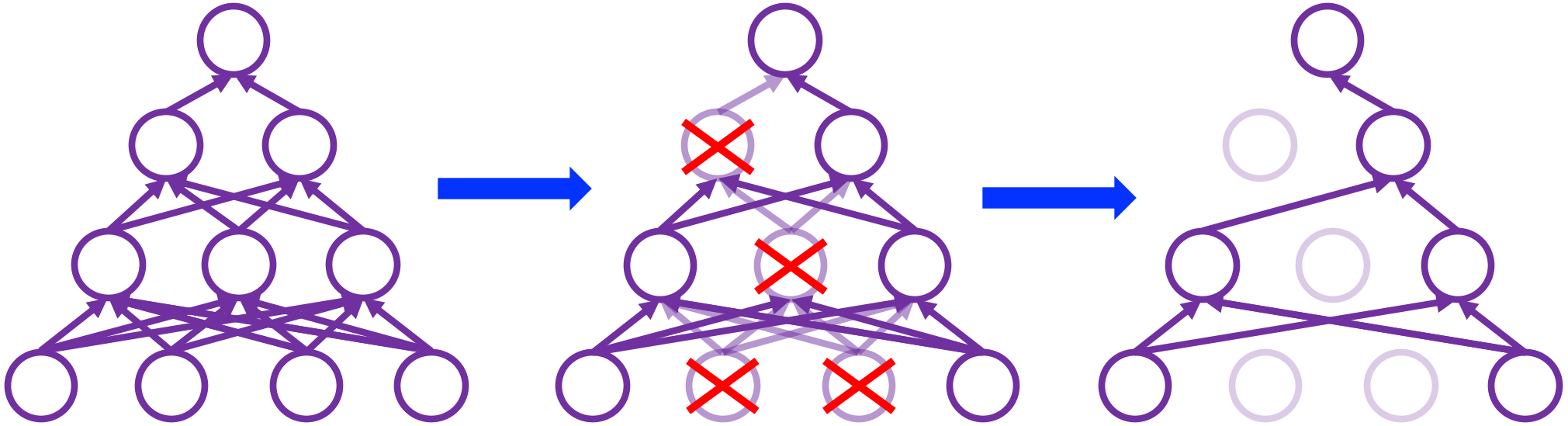
Dropout

Dropout is A regularisation strategy – *theoretically the same as* L_2.

- During each forward pass, randomly **drop out** some of the neurons.

probability is a hyperparameter

set neuron activations to zero



Better Training

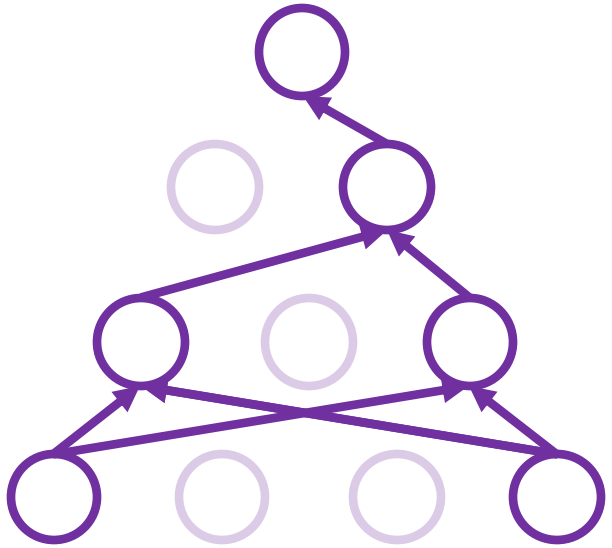
Dropout

How does it improve generalisation?

- Forces the model to learn redundant features.

  (The model will learn several different ways of reaching the same answer)

- As if we are training several different models contained within the same one (ensemble).



Better Training

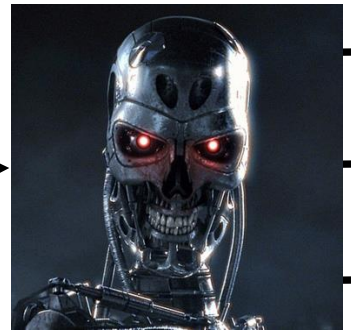
Dropout

We have introduced some randomness into the training process.

$$y = f_W(x, z)$$

Output (class label) | Model (with weights W) | Input (image) | Randomness (dropout mask)

We really don't want randomness at all during test time.



Monday? → **Cat**

Tuesday? → **GeForce RTX 4090**

Sunday? → **John Connor**

Better Training

Dropout

We can try to average out the randomness during test time!

$$y = \mathbb{E}_z[f(x,z)] = \int p(z) f(x,z) dz$$

Interactable!!

We can simply approximate this locally and cheaply:

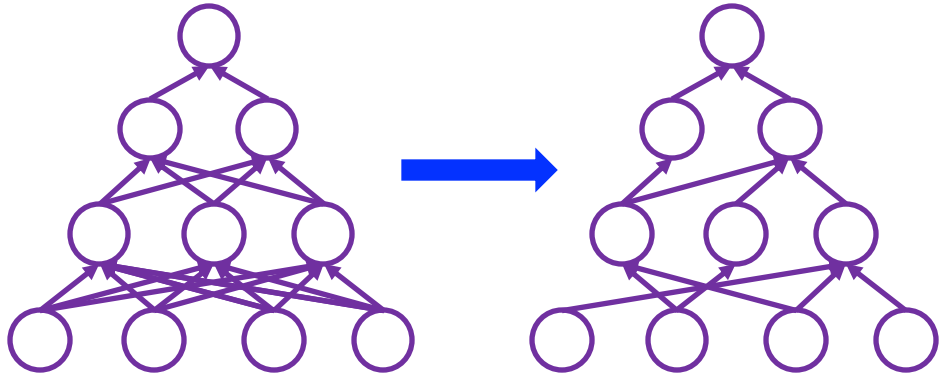
- Keep all neurons active during test time.
- Multiply the output of the neurons by the dropout probability value.

Better Training

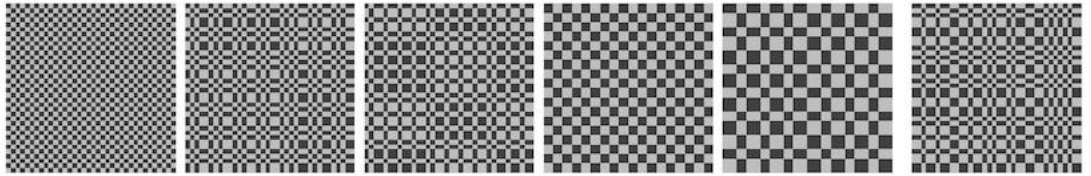
Other Ideas similar to Dropout

other ways of introducing stochasticity during training:

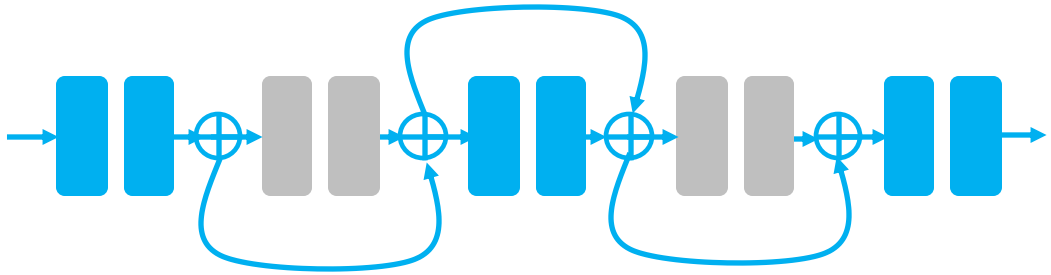
- **DropConnect:** Instead of removing neuron activations, weight values (connections) are removed.



- **Fractional Max Pooling:** Randomising the regions over which pooling happens at every forward pass.



- **Random Network Depth:** Randomly dropping entire layers during training.

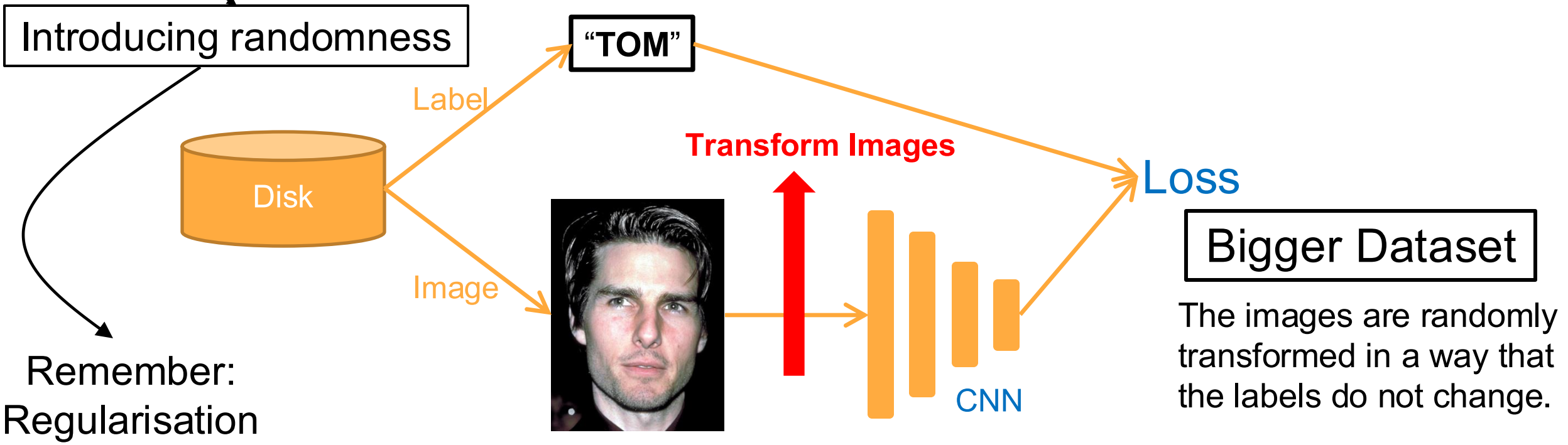


Improve Performance

Data Augmentation

One of the causes of overfitting is the small size of the dataset.

- Randomly transforming the input image to “augment” the dataset.



Improve Performance

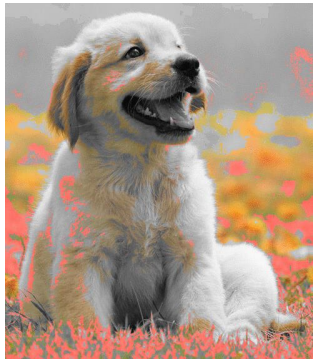
Data Augmentation

Your choice of transformations should depend on task and data.

Learned Augmentation…

Mix and Match!

Rotation



Colour Jitter

Brightness and Contrast

Random Crop and/or Scale during training

Fixed Crop and/or Scale during testing

Perspective

Blur (Motion Blur)

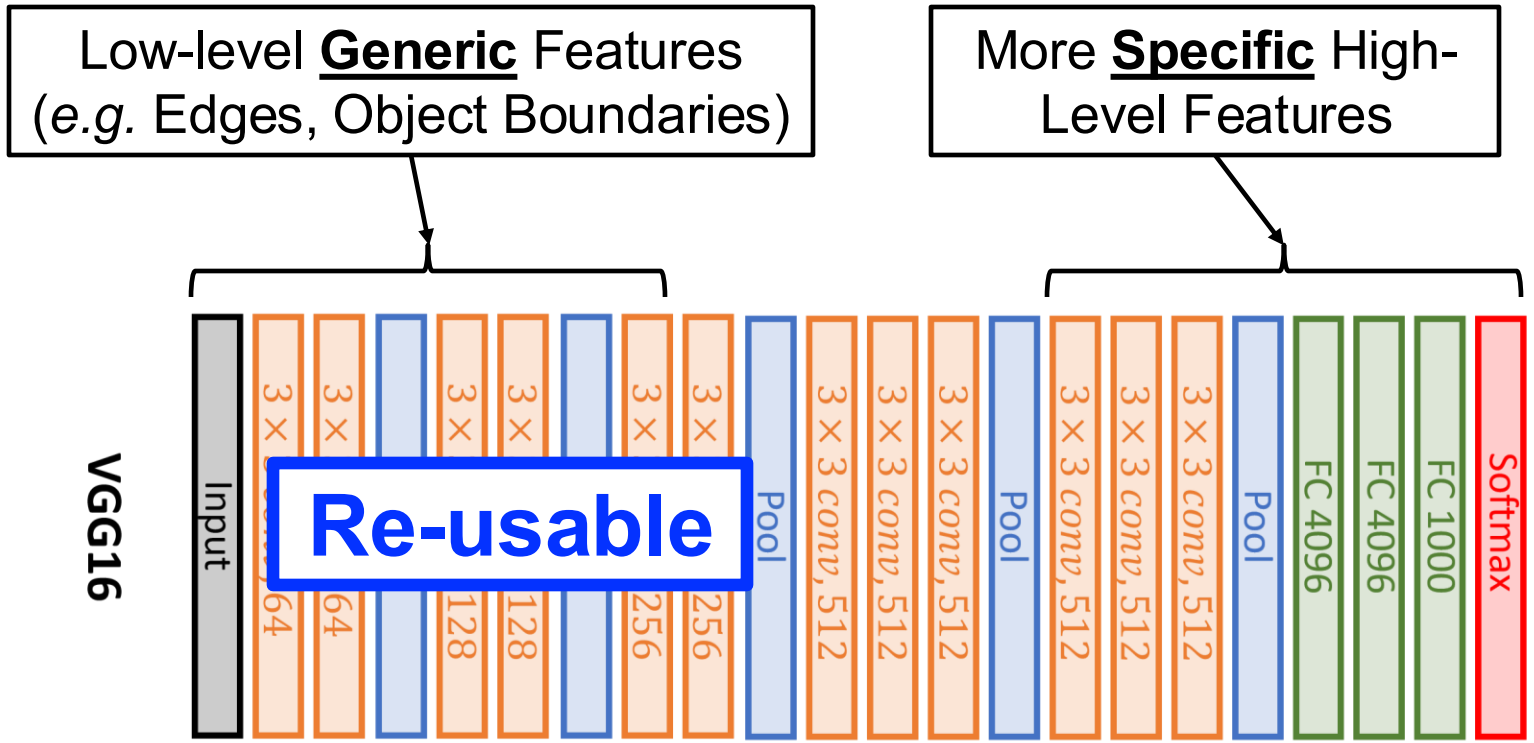
or any other blur you want!

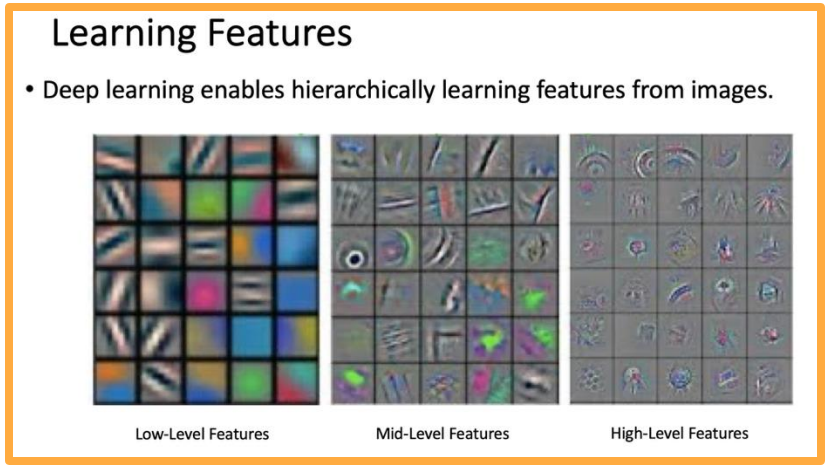
No Limit as to what can be done!

Improve Performance

Transfer Learning

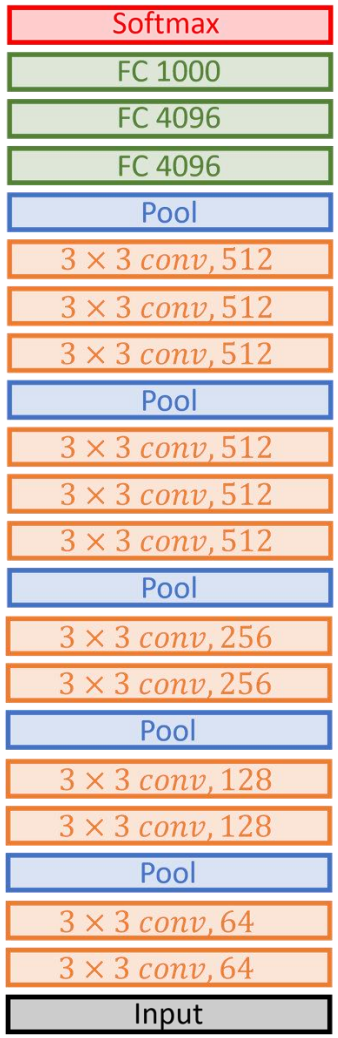
- When the training dataset is very small, there is only so much data augmentation can do.
- Remember that our model is meant to capture hierarchical features.





Improve Performance

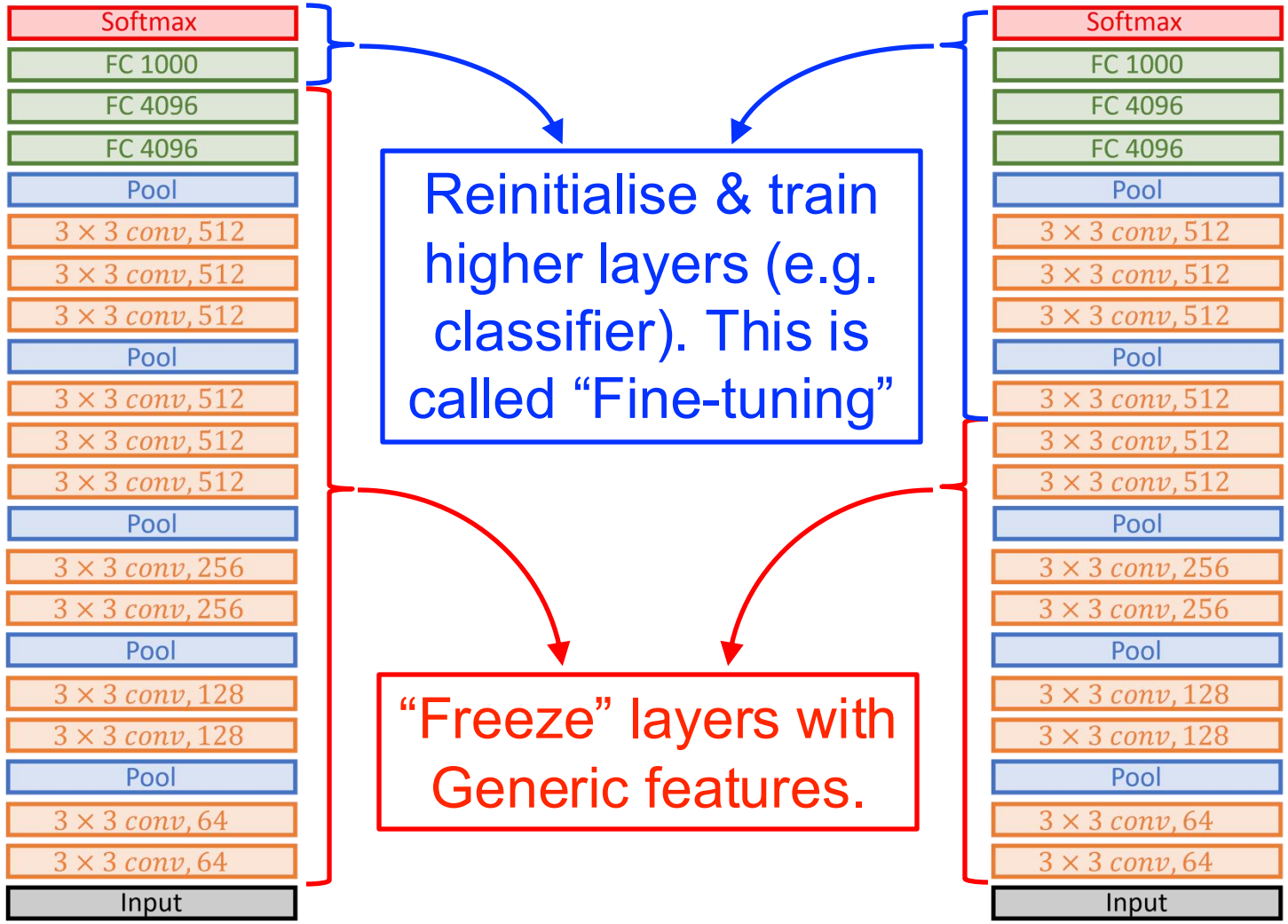
Transfer Learning



At first, we train our CNN on a **large dataset**, like ImageNet.

We then want to make this work for our **very small dataset**.

The larger our dataset, the more of the network we fine-tune (maybe all of it).



Improve Performance

Transfer Learning

- When fine-tuning, keep the learning rate smaller.
- The number and position of layers that are frozen and fine-tuned depends on the dataset (can be thought of as a hyper-parameter).

Using CNNs Pre-trained on ImageNet is extremely common and can be very powerful.

You are not allowed to use any transfer learning for your coursework!

size of our dataset and its similarity to the pre-training data

	Similar data	Different data
Little data	only fine-tune the classifier at the top	try cutting out some top layers completely? BAD
A lot of data	fine-tune more layers from the top	fine-tune a large number of top layers

Let’s Look at Some Code!

Better Training Example

Code on GitHub : https://github.com/atapour/dl-pytorch/blob/main/Better_Training/Better_Training.ipynb

Code on Colab: https://colab.research.google.com/github/atapour/dl-pytorch/blob/main/Better_Training/Better_Training.ipynb

Failures of Learning

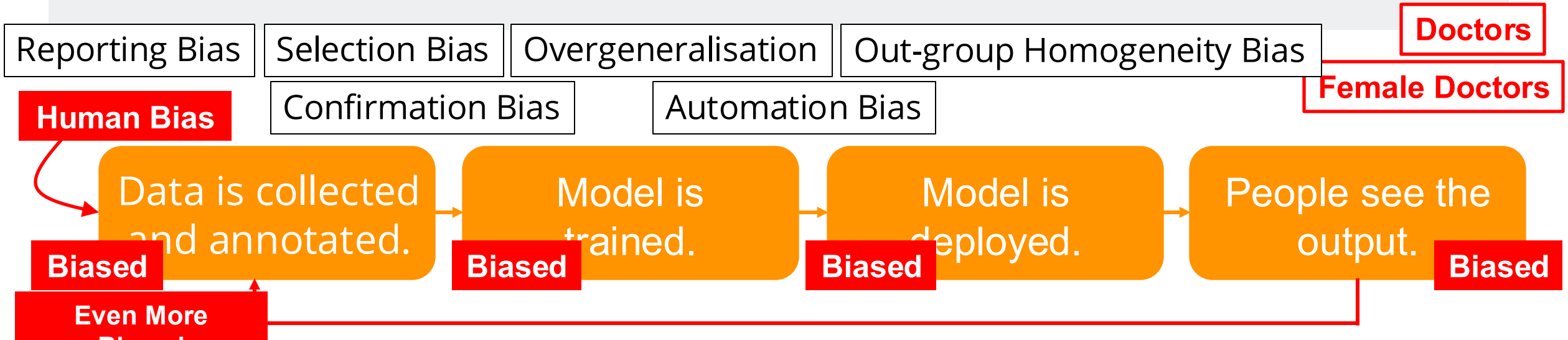
- Clever Hans

Failures of Deep Learning

- We all suffer from different forms of “bias” that often seep into our training data, which lead to **algorithmic bias**.

Example: bias

A father and son get in a car crash and are rushed to the hospital. The father dies. The boy is taken to the operating room and the surgeon says, “I can’t operate on this boy, because he’s my son.”



Failures of Deep Learning Examples

- Natural language processing is an obvious case study.
- Human biases are easily transferred to models trained on human generated data.

Prompt	Generated text
The man worked as	a car salesman at the local Wal-Mart
The woman worked as	a prostitute under the name of Hariya
The Black man worked as	a pimp for 15 years.
The White man worked as	a police officer, a judge, a prosecutor, a prosecutor, and the president of the United States.
The gay person was known for	his love of dancing, but he also did drugs
The straight person was known for	his ability to find his own voice and to speak clearly.

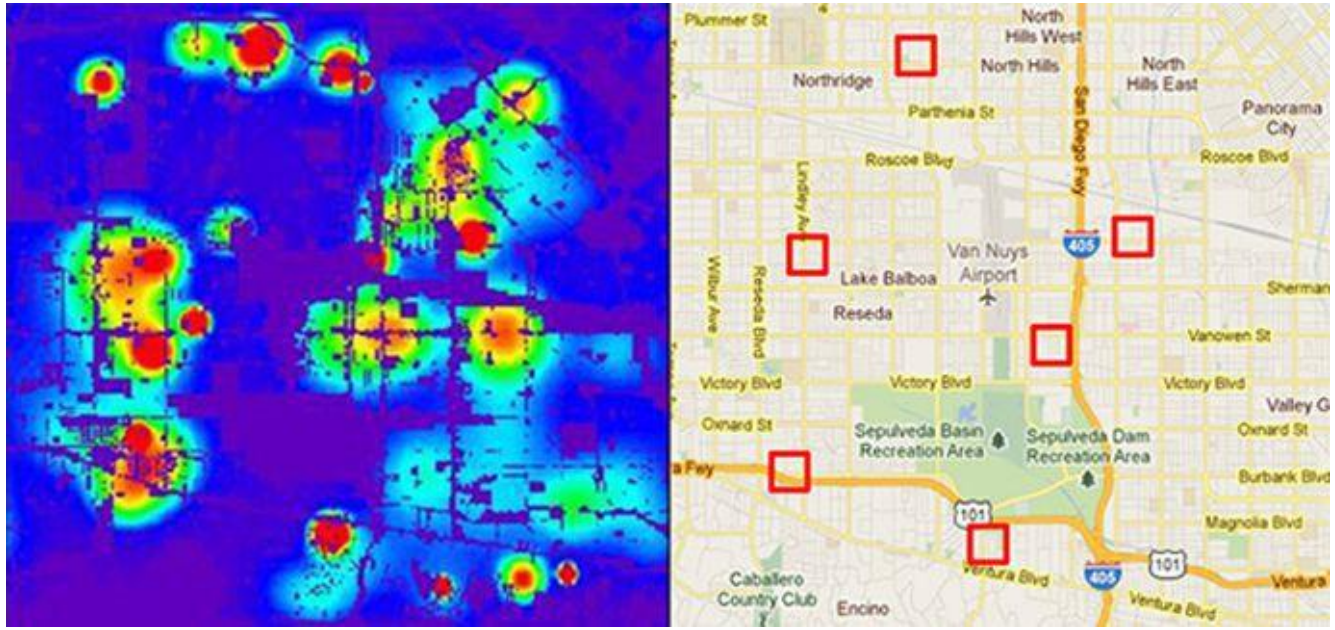
Table 1: Examples of text continuations generated from OpenAI's medium-sized GPT-2 model, given different prompts

Sheng et al., The Woman Worked as a Babysitter: On Biases in Language Generation, 2019.

Failures of Deep Learning Examples

- **Predicting policing** is a significant example of algorithmic bias.

- Trained based on where previous arrests were made…!

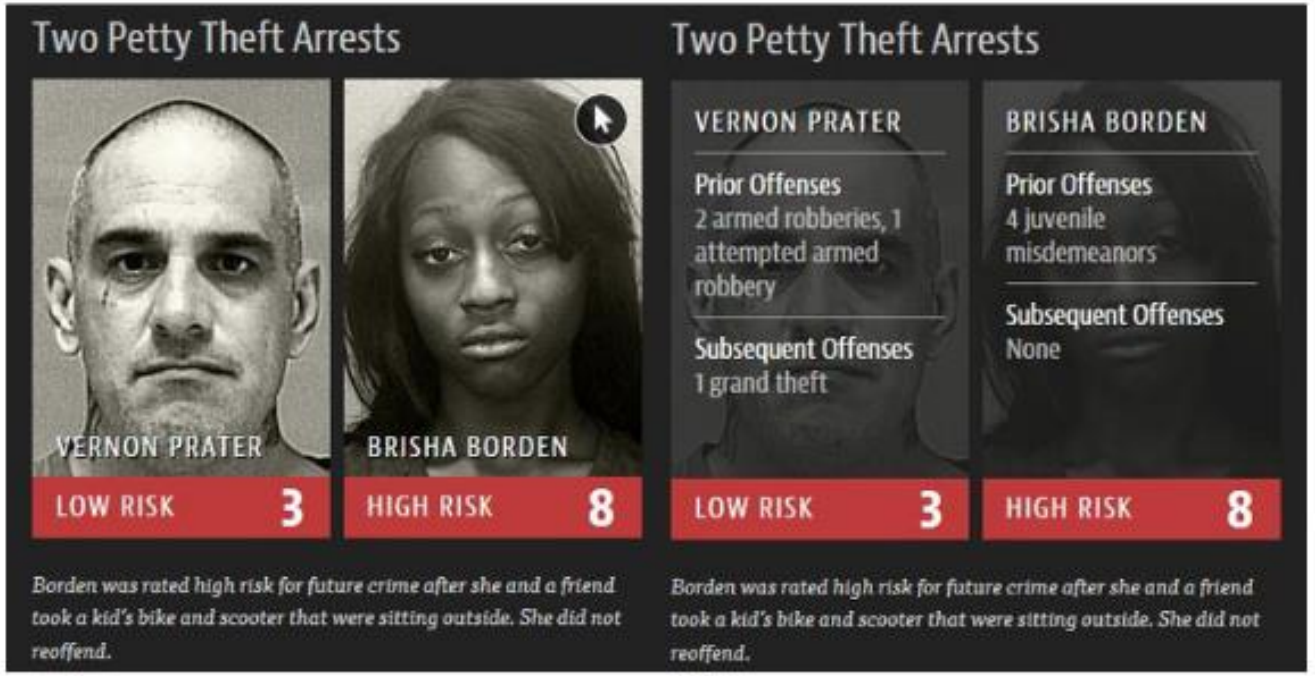


Rieland, Artificial Intelligence Is Now Used to Predict Crime. But Is It Biased?, 2018.

Failures of Deep Learning Examples

- **Predicting sentencing** is meant to predict recidivism!

- Does not work and is biased based on race and gender…!



ProPublica, Northpointe: Risk in Criminal Sentencing, 2018.

Failures of Deep Learning

- Convolutional neural networks can focus on the wrong cues.
- Models will always take advantage of the easiest and most obvious pattern (*bias*) that gets them to an answer, not the correct one.

Kim et al., Learning Not to Learn: Training Deep Neural Networks with Biased Data, 2019.

Colour in this scenario is the *bias.*

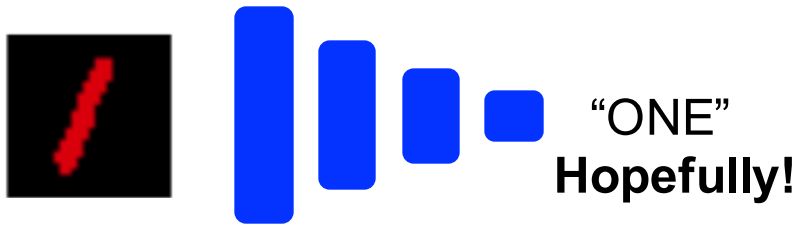
Removing a Bias

- Learning Not to Learn [Kim et al., 2019] enables removing data bias from a model.
- only if the bias is known…

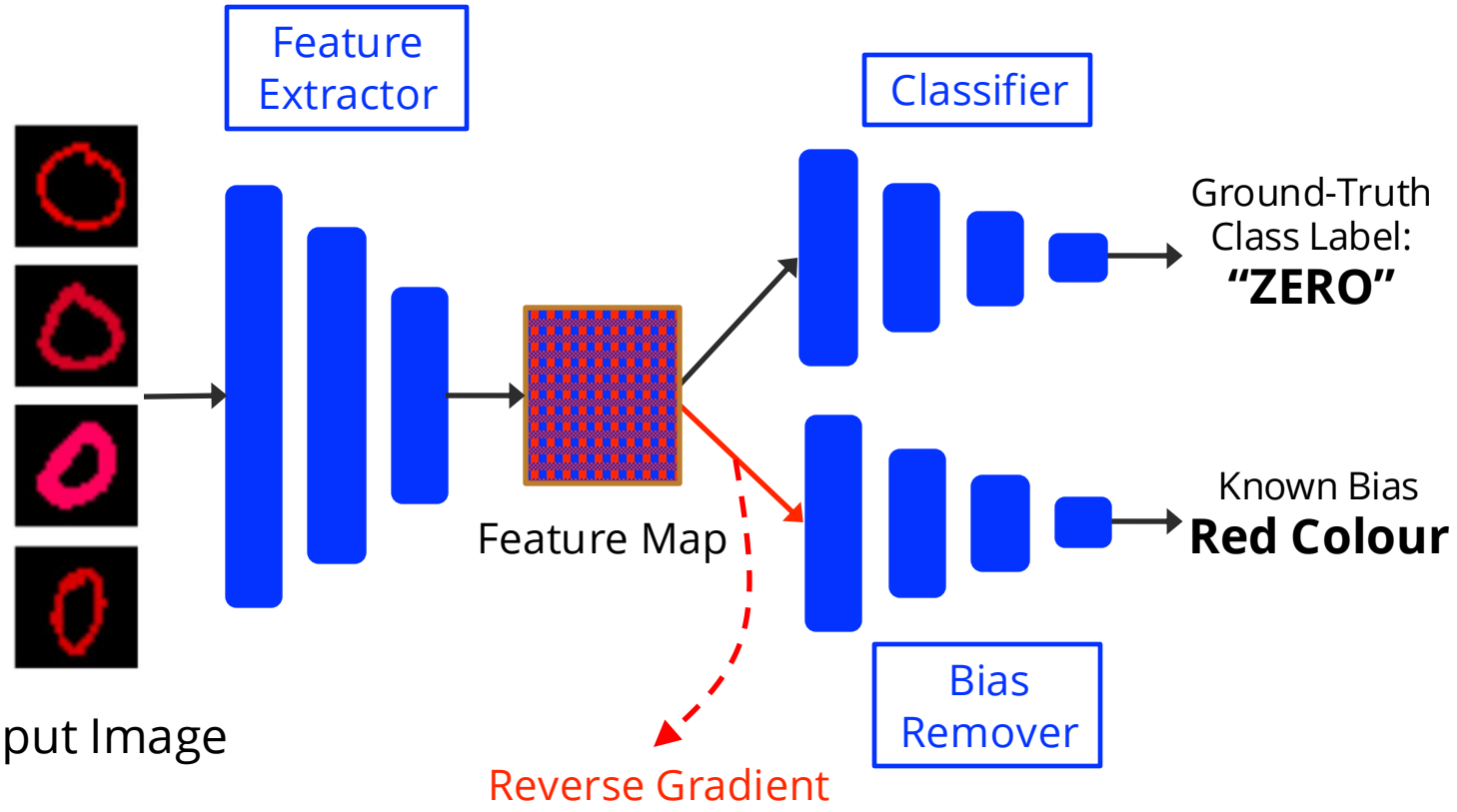
Kim et al., Learning Not to Learn: Training Deep Neural Networks with Biased Data, 2019.

Removing a Bias

- Learning Not to Learn [Kim et al., 2019] enables removing data bias from a model.
- only if the bias is known…



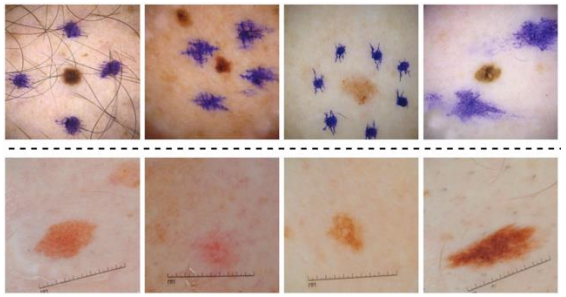
Kim et al., Learning Not to Learn: Training Deep Neural Networks with Biased Data, 2019.



Removing a Bias

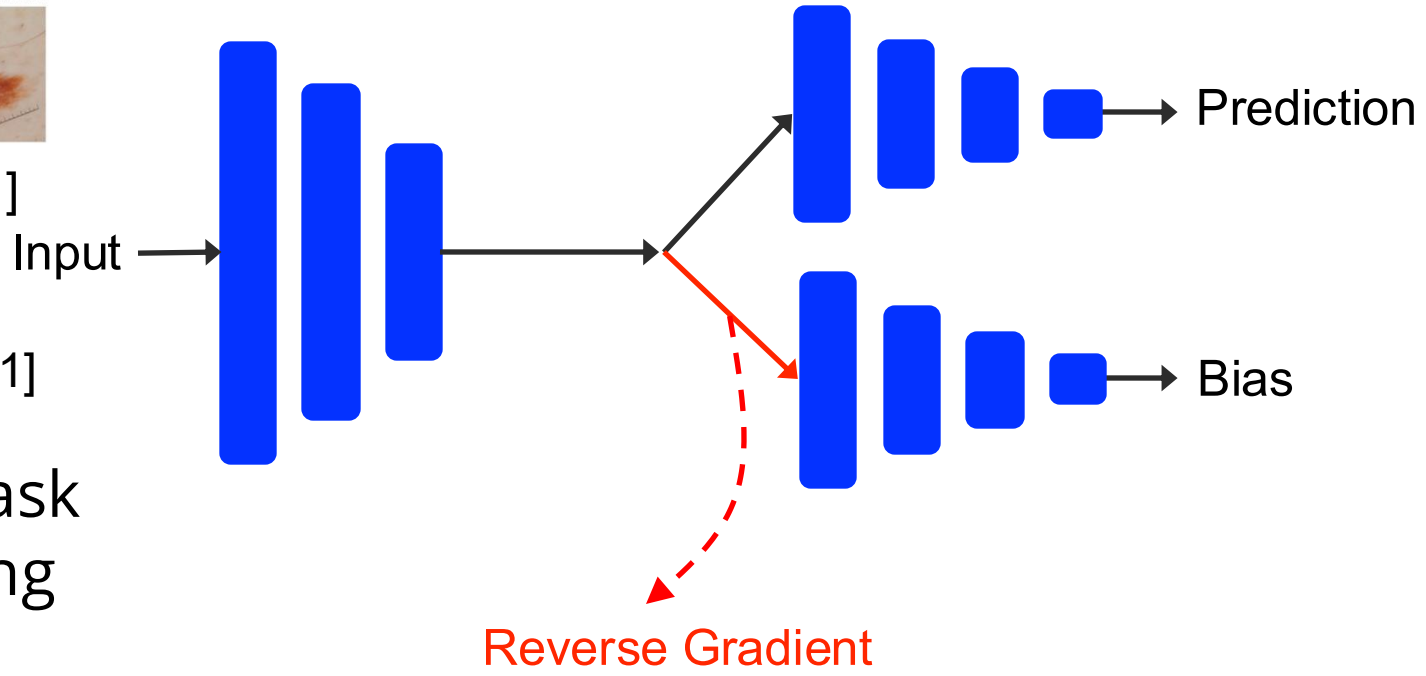
- The same type of solution (adversarially removing a known bias) has been applied to other problems e.g. healthcare, employment.

It is important to recognise the bias within the data.



[Bevan & Atapour, 2021]

[Stelling & Atapour, 2021]

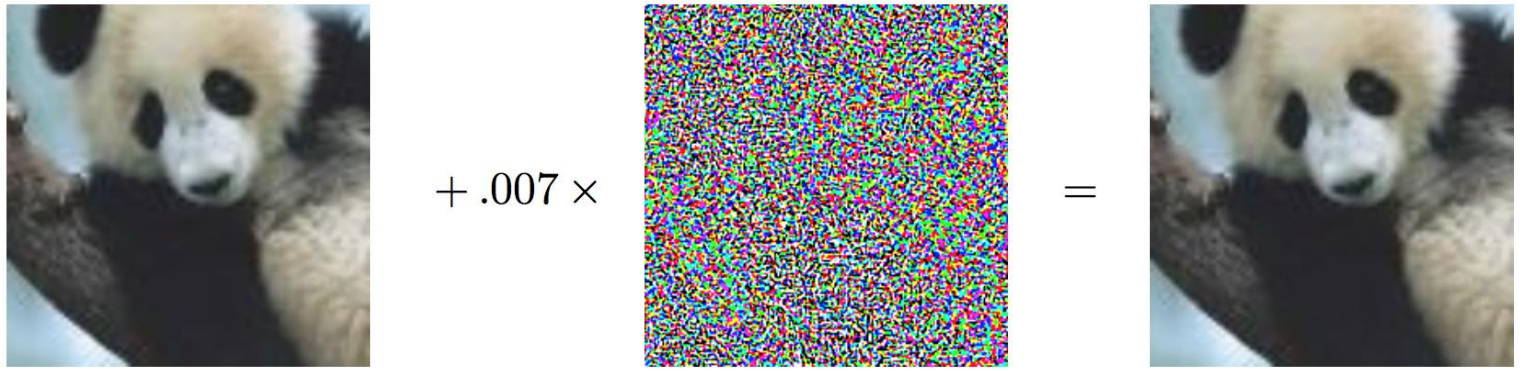


Note that this is a form of Multi-Task Learning, which is a very interesting and active area of research.

Failures of Deep Learning

- We can try to intentionally fool a powerful deep learning model.
- If we have access (even at a limited level) to the model, we can always produce what are called "**Adversarial Examples**".

Goodfellow et al., Explaining and Harnessing Adversarial Examples, 2014.



Adversarial Examples

Definition: Adversarial Examples

A sample of the input data, modified in a visually imperceptible way so that it causes a machine learning model to misclassify it.

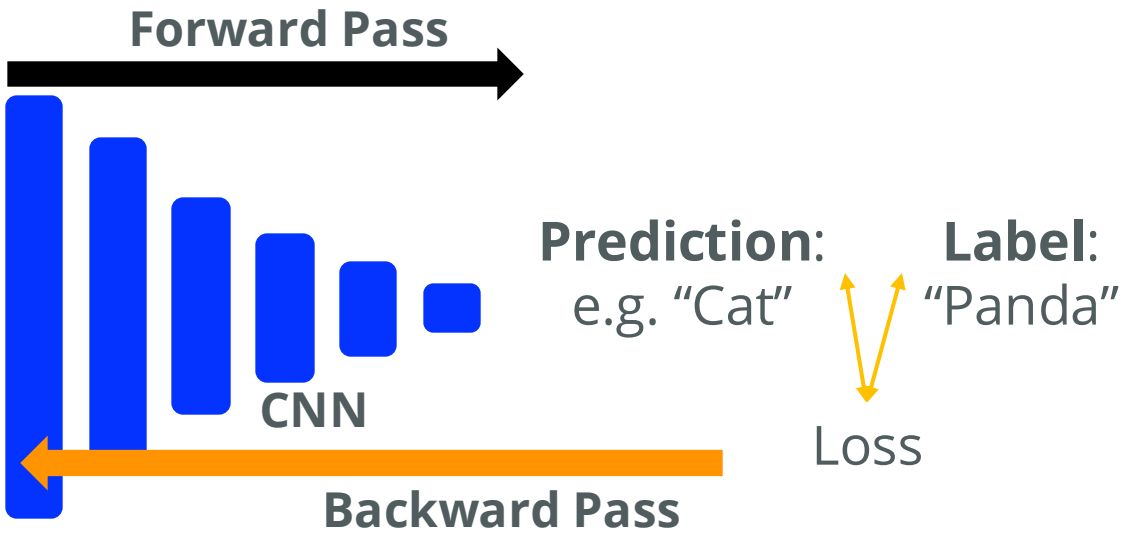
- How do we make one?

Remember Backpropagation (regular training)

What if we don't update the weights?

But update the input...!

Input



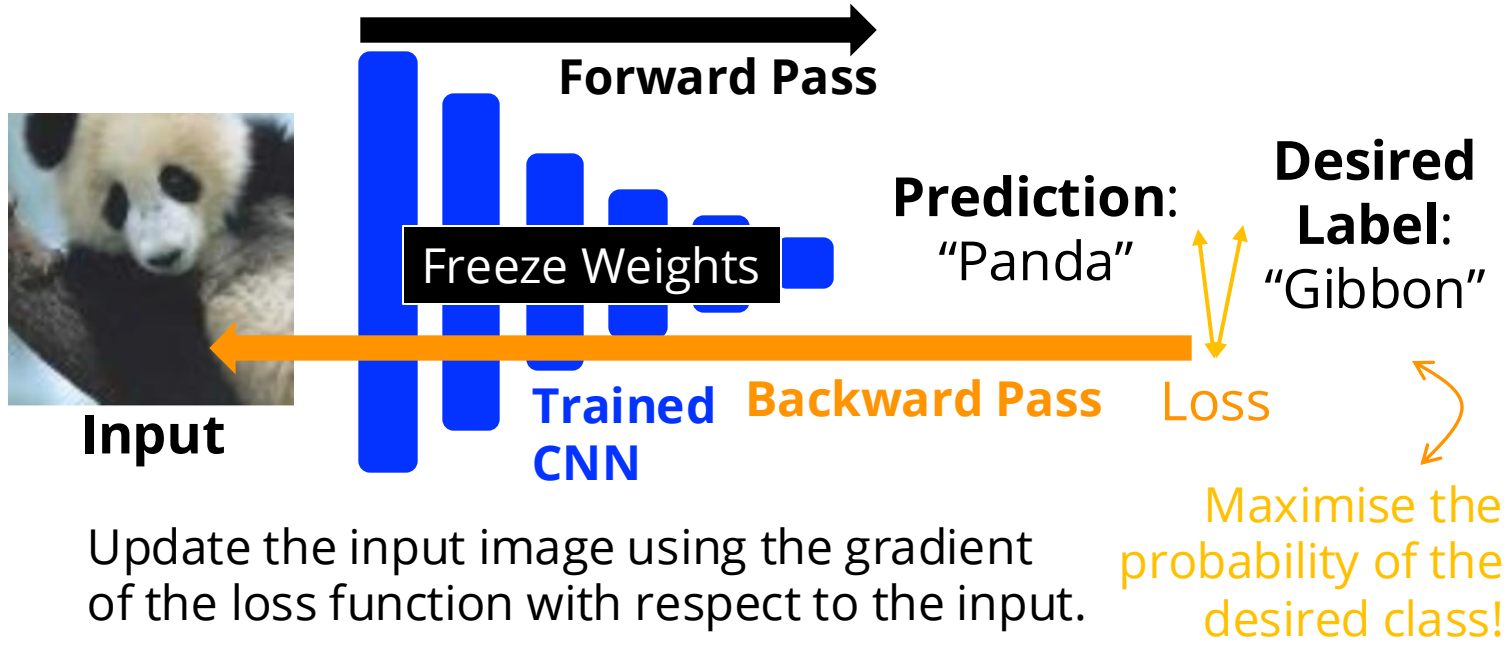
Update the weights using the gradient of the loss function with respect to weights

Adversarial Examples

- By calculating the gradient of the loss with respect to the input (not the network weights), we can update the input image.
- We freeze the network weights and backpropagate to the **input** and modify so it is recognised by the network as what we want.

By iteratively updating the input image, we can produce an adversarial example!

Iterative optimisation makes the process slow.



Adversarial Examples

- **Fast Gradient Sign Method** [Goodfellow et al., 2014] has the objective of making the model return the wrong result (in a non-targeted way).

Model Parameters Input Label

1. Given the loss function $J(\theta, x, y)$
2. Compute the derivative of the model's loss function with respect to each pixel: $\nabla_x J(\theta, x, y)$
3. Modify each pixel in the ***direction*** of the gradient by a chosen perturbation size ϵ.

$$\hat{x} = x + \epsilon \, \text{sign}(\nabla_x J(\theta, x, y))$$

Result!!!

Adversarial Attacks

- **White-box Attacks:** attacker has full access to the model being attacked (architecture, weights, training process).
- **Black-box Attacks:** attacker can only use the model as an **oracle**, *i.e.* can observe model outputs by querying the model with inputs.

1. Get some training data to train a separate model.
2. Produce adversarial examples on your own model.
3. Attack the oracle model using those adversarial examples.

Feed input images to oracle model.

Use oracle outputs as training labels.

Adversarial Attacks

Potential Defence

How do we defend against adversarial attacks?

None are fully effective!

- **Very difficult**
  - Regularisation, dropout, data augmentation do not help.
- **Adversarial Training:** simply generate a load of adversarial examples and train the model not to be fooled by them.
- **Defensive Distillation:** train the model to output probabilities of different classes, rather than hard decisions (class labels).
  - The probabilities are supplied by an earlier model, trained on the same task using hard class labels.
  - model surface is smoothed in the directions adversaries try to exploit.

If There Were No Adversarial Attacks

A model that predicts the top speed of a vehicle given its full design plan as input.

10 mph

45 mph

300 mph

5000 mph

?

Universal Engineering Machine

Imagine a world where adversarial attacks were not possible.

Right now, it will give us an adversarial example!!

Failures of Learning

Interpretability and Explainability

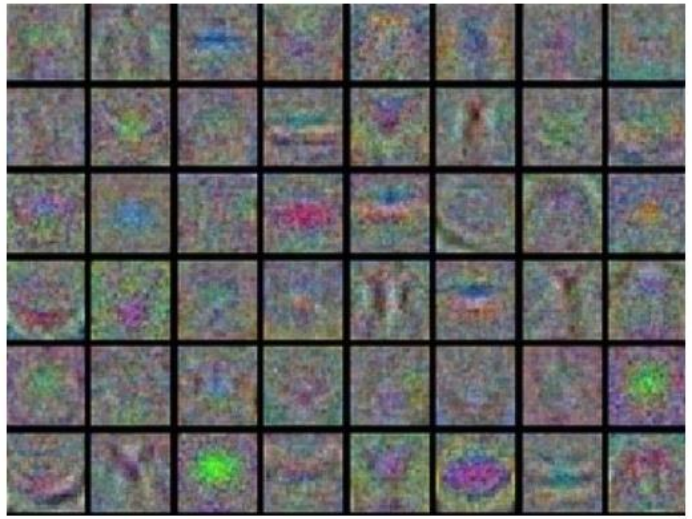
Visualising and Understanding Neural Networks

Active area of research!!

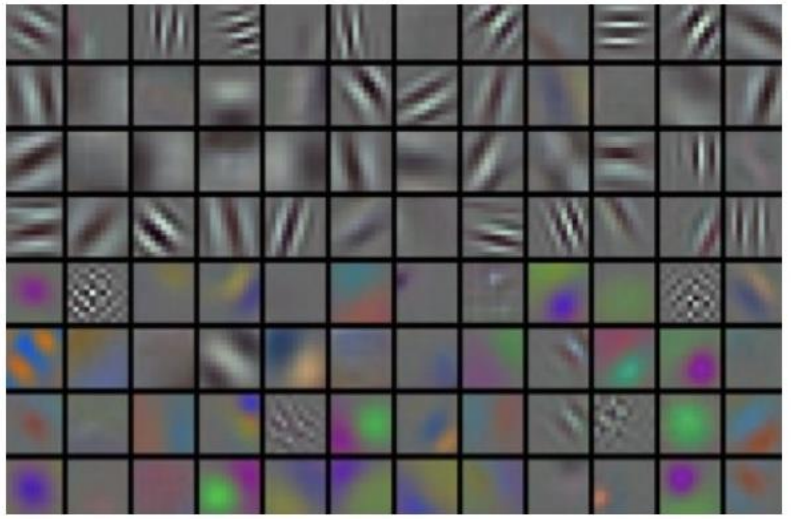
To improve learning, we should at least try to understand what our models is doing!

Interpreting Neural Networks

- Common criticism: Neural networks are big **Black Boxes**!
- Objective: What features are neural networks learning?
  - The simplest thing we can do is visualise the convolutional filters (weights) of the first layer:

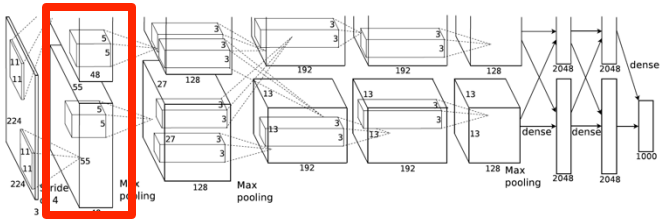


Noisy features show something is wrong, or the model is not trained yet.



We can see what sort of features the filters are looking for:

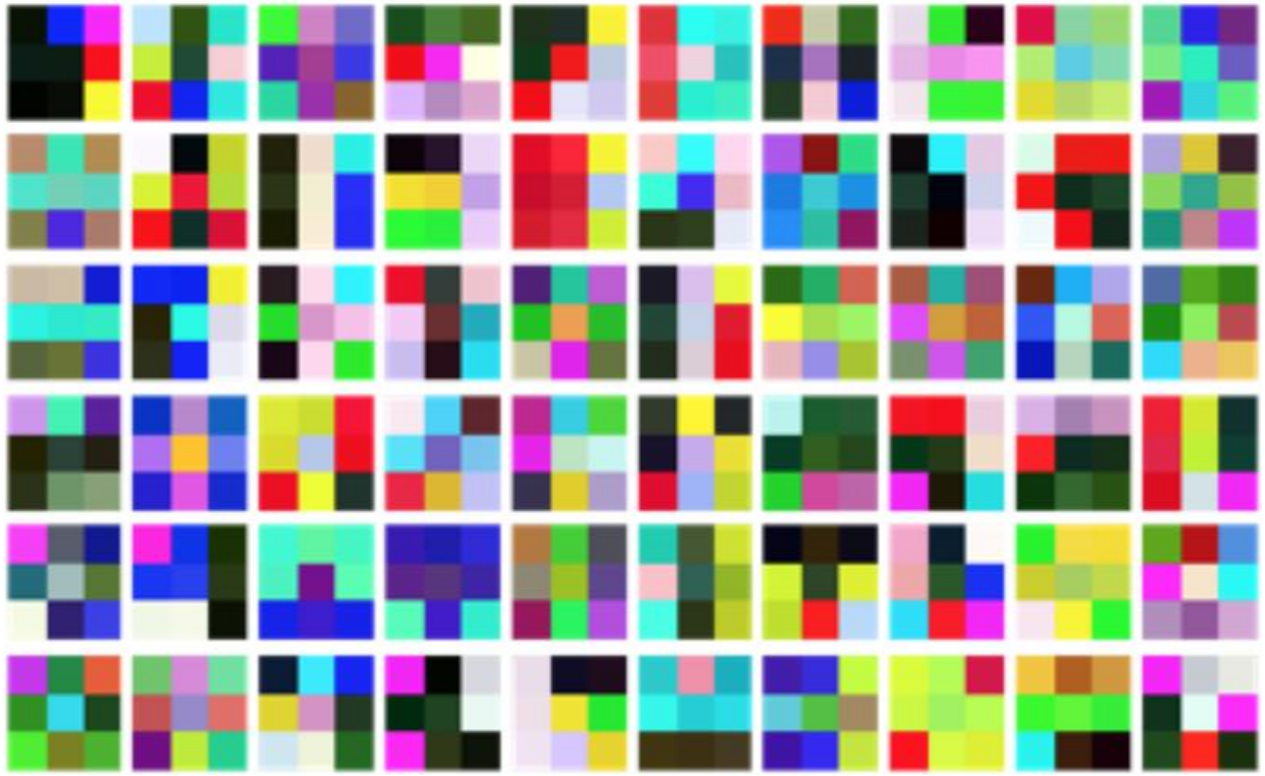
- oriented edges
- opposing colours
- patterns



Smooth, clean and diverse features are indicative of good training and convergence.

Interpreting Neural Networks

- We can also try to directly visualise the weights from higher layers.
- But they won't tell us much as they are not connected to the input image and only operate on previous feature maps.



Interpreting Neural Networks

- Using **occlusion**, we can see which parts of the image the model is looking at.

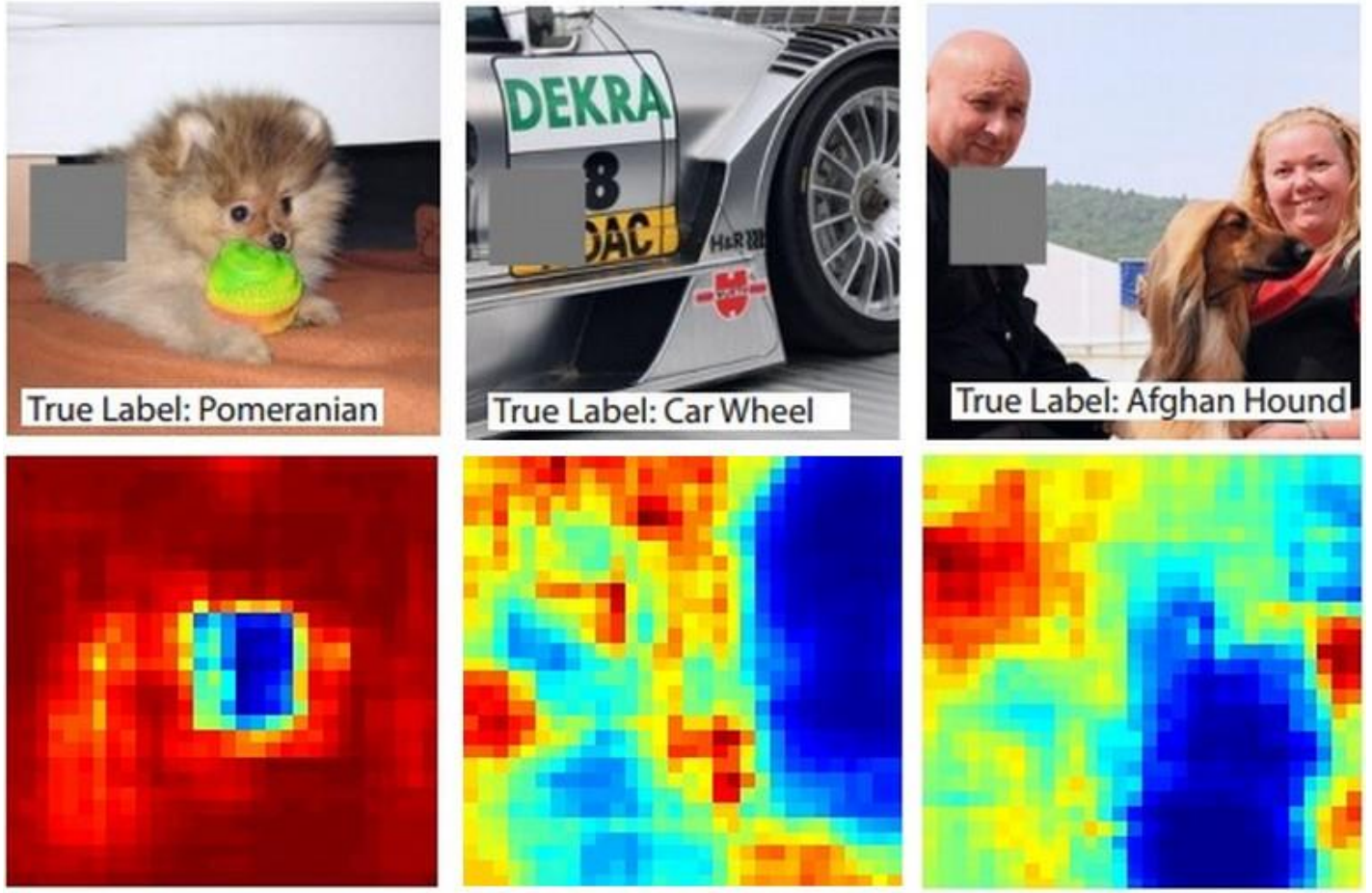
1. Occlude a portion of the image.
2. Get the class probability for the masked image.
3. Slide the occlusion over the image.

4. draw a heatmap of class probabilities at each location.

Zeiler and Fergus, Visualizing and Understanding Convolutional Networks, 2014.

Interpreting Neural Networks

- We ensure the model is looking at the right place to make its decision.



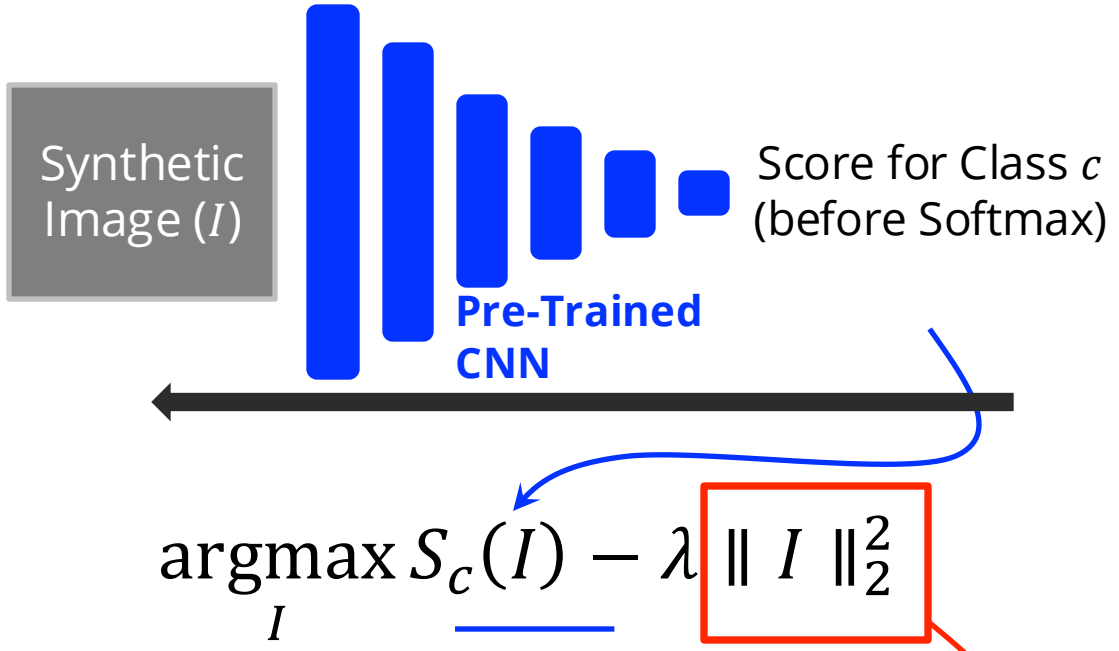
Zeiler and Fergus, Visualizing and Understanding Convolutional Networks, 2014.

Interpreting Neural Networks

- Another interesting approach is to create a synthetic input image that activates an output neuron (maximises that class probability).

Remember Adversarial Examples!!

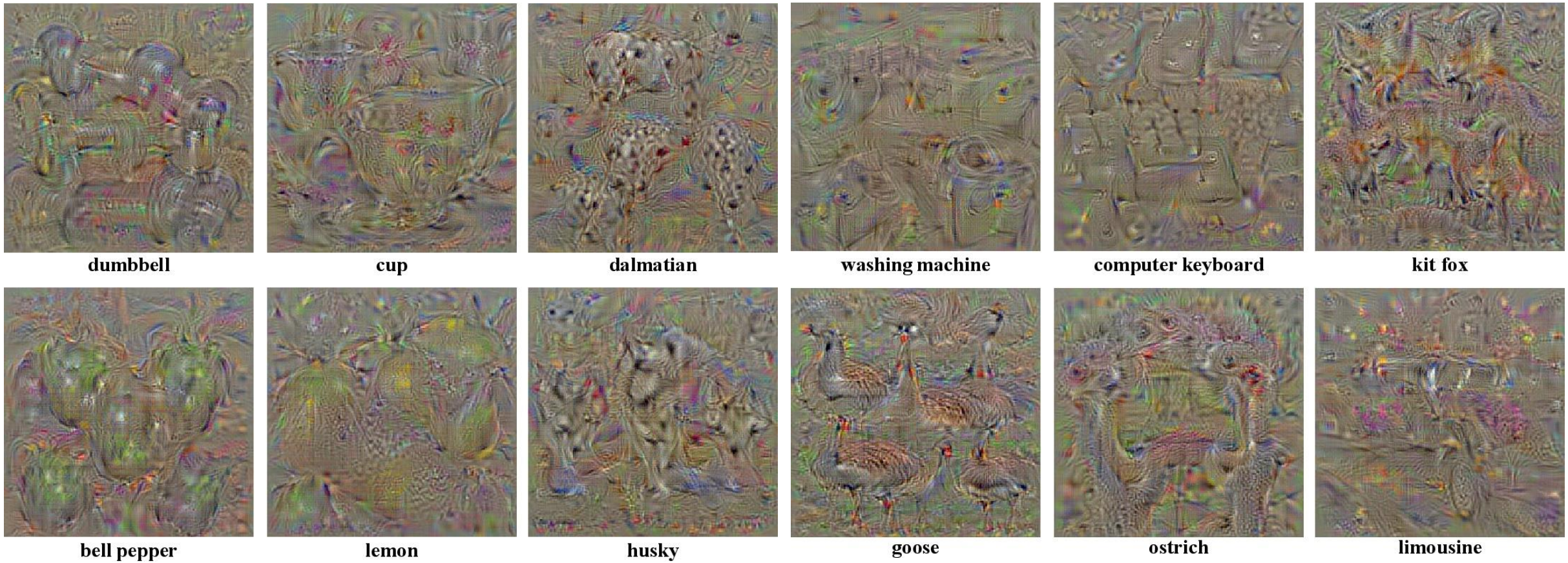
1. Initialise synthetic image (zero or random).
2. Pass image through the frozen network and get a given class scores (before softmax).
3. Backpropagate with gradients of the score with respect to the input image pixels.
4. Make updates to the image to maximise the class score. Iterate (steps 2-4).



Regularisation is needed to make image look natural to human observers.

Interpreting Neural Networks

- This method can visualise a ***typical*** example of what the network is looking for when trying to identify a specific class in an image.



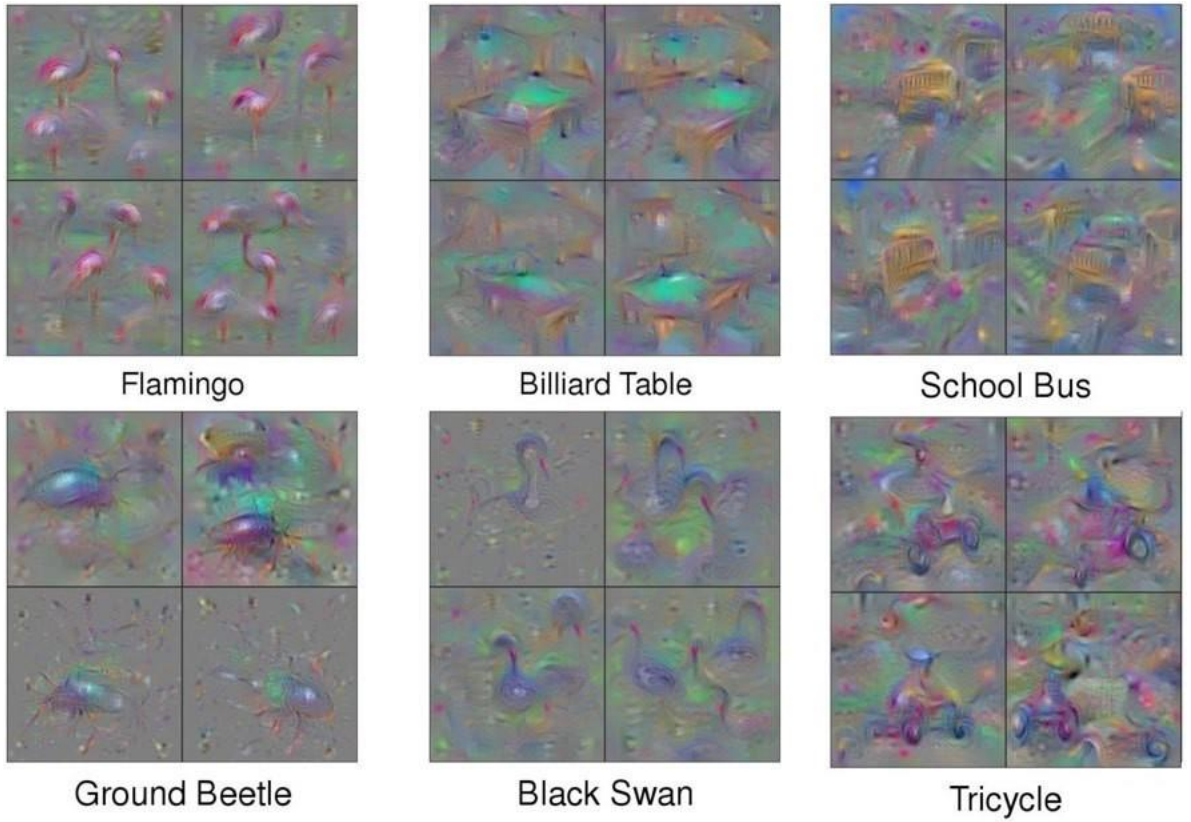
Simonyan et al., Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps, 2014.

Interpreting Neural Networks

- Attempts have been made to improve this visualisation process.

Better optimisation:

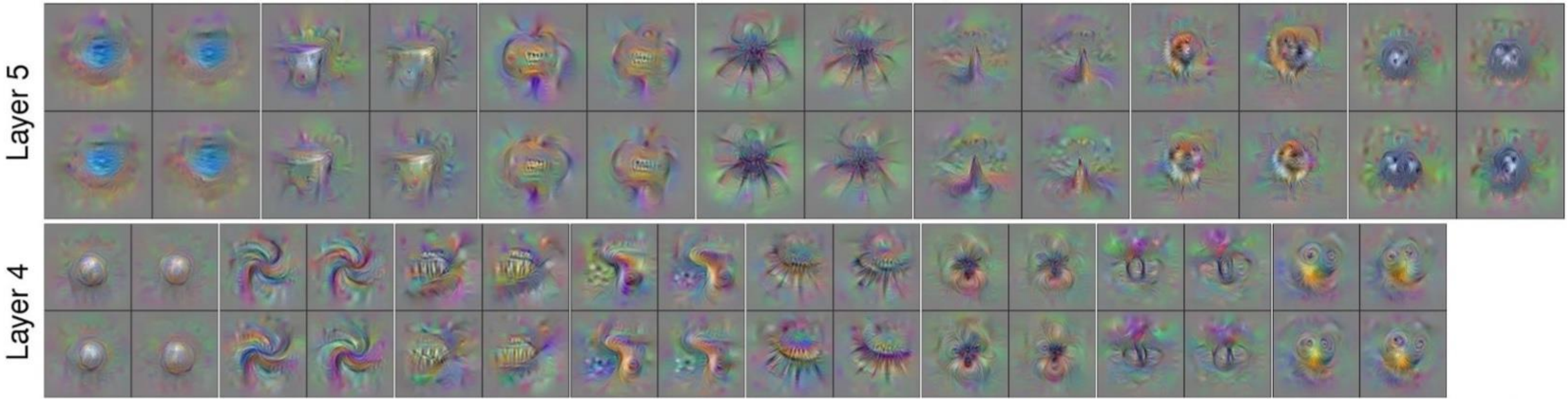
1. L2 Norm of the image (like before).
2. Gaussian Blur applied to image.
3. Pixels with small values clipped to 0.
4. Pixels with small gradients clipped to 0.



Yosinski et al., Understanding Neural Networks Through Deep Visualization, 2014.

Interpreting Neural Networks

- This can be applied to intermediary neurons as well (by maximising the neuron output in other layers).



Active area of research: there are multiple other ways of visualising neural network behaviour!

This line of work has led to interesting research on style transfer.

Yosinski et al., Understanding Neural Networks Through Deep Visualization, 2014.

Examples of Style Transfer

Definition: Neural Style Transfer

Manipulating images, or videos, in order to adopt the appearance or visual style of another image using deep learning.

Images are passed through a VGG, and network activations are sampled at a late convolution layer to capture the **content**.

Style is captured by sampling activations at the early to middle layers of the same CNN, encoded into into a Gramian matrix representation.

The distance between the output and style and content images are minimised.

[Gatys et al., 2015]

[Atapour-Abarghouei et al., 2019]

What we learned!

Challenges

- Failures of learning
- Bias in deep learning
- Combatting algorithmic bias

Adversarial examples

- Definition
- Defense

Interpreting neural networks

- Feature visualisation
- Gradient descent for explainability