DEEP LEARNING

- ImageNet Challenge & Breakthrough Deep Neural Networks
    1. ImageNet (Dataset & Challenge)
    2. AlexNet (first DNN winner of ImageNet)
    3. VGGNet (Deeper DNN, runner up in ImageNet)

- Going Deeper: Is this the solution?

- ResNet: the solution

- References

ImageNet Dataset

- large annotated photographs' dataset for computer vision research
- goal: resource for promoting research and development of improved methods for computer vision [1]

ImageNet Large Scale Visual Recognition Challenge (ILSVRC)

- annual competition held between 2010 and 2017 [2]

- challenge tasks use subsets (approximately 1.2 million images) of the ImageNet dataset for:

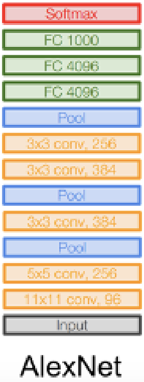
i) "*image classification*": assigning a class label to each image based on the main object in the photograph (among 1,000 object classes)

ii) "*object detection*": localizing the objects within each photograph

AlexNet: First Deep Neural Network Winner of ILSVRC 2012

- In 2012, AlexNet [3] significantly outperformed all prior competitors (error 15.3%; prior competitors' error was 25.7% and 28.2%)

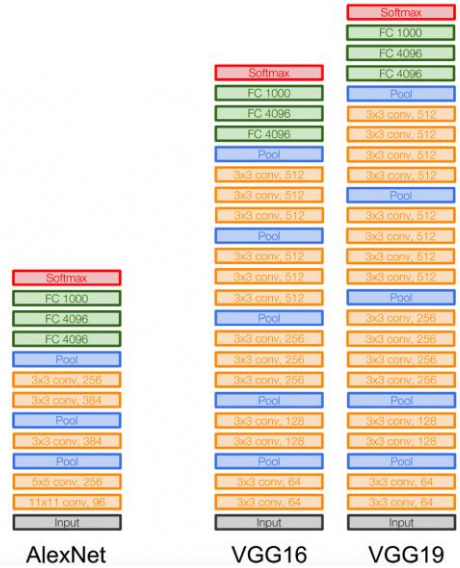
- The runner up was not a deep learning method (error 26.2%)



VGG: Runner-Up of ILSVRC 2014, Deeper than AlexNet

- was the runner-up at the ILSVRC 2014
- achieved error 7.3% (vs 15.3% of AlexNet)
- has 16 or 19 layers and is deeper than AlexNet (8 layers)
- however, VGG [4] consists of 138 million parameters (AlexNet consists of 61 million parameters)

VGG & AlexNet Architectures



VGG/AlexNet: Conclusions

Should we make a Neural Network (NN) deeper and why?

more layers → more high-level features → better understanding of data and better prediction

Neural Networks → make them deeper → problem solved?

If yes, how deep?

- ImageNet Challenge & Breakthrough Deep Neural Networks
- ➢ Going Deeper: Is this the solution?
  i. Issues to consider
  ii. Specific Problems:
     - Vanishing Gradients
     (definition, cause, significance, comparison shallow & deep NN, solutions)
     - Degradation
     (definition, analogy, not overfitting)
- ResNet: the solution
- References

Depends on:

- The complexity of the task at hand
- Available computational capacity during training
- Available computational capacity during inference

If the task needs a lot of parameters:

- Can we train very deep networks efficiently using current optimisation solvers?
- Is training a better model as simple as adding more and more layers?

1. Vanishing Gradients

1) Vanishing Gradients: the problem

- During each iteration of standard neural network training, all weights receive an update proportional to the partial derivative (gradient) of the cost function with respect to their current value
- If the gradient is very small then the weights will not change effectively
- As a consequence, this may completely stop the neural network from further training
- This is called the vanishing gradient problem.

Vanishing Gradients: the causes

The Vanishing Gradient Problem is met in Neural Networks:

- with certain activation functions
- trained with gradient based methods (e.g Back Propagation [5])

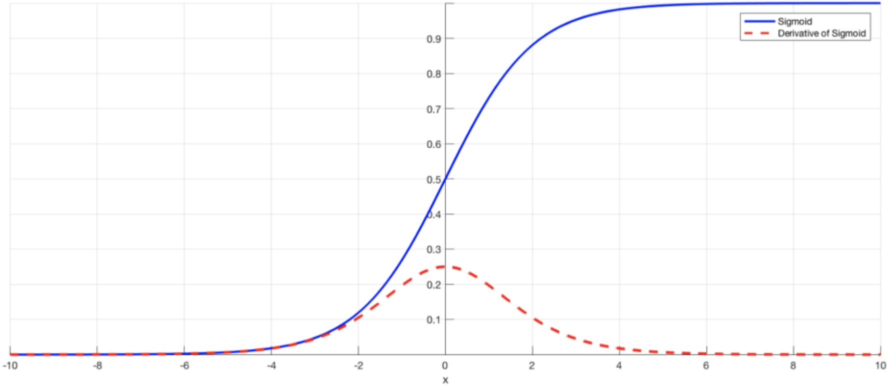
It gets worse as the number of layers in the neural network increases.

Vanishing Gradients: caused by activation function (1)

Vanishing gradient problem depends on the choice of the activation function:

- many common activation functions (e.g., sigmoid [6], tanh [7]) 'squash' their input into a very small output range in a very non-linear fashion
- for example, sigmoid maps the real number line onto a "small" range of [0, 1] → large regions of the input space are mapped to an extremely small range
- in these regions of the input space, even a large change in the input will produce a small change in the output - the gradient is small.

Vanishing Gradients: caused by activation function (2)



Vanishing Gradients: caused by gradient descent training

Gradients of neural networks are usually computed using backpropagation:

- backpropagation finds the derivatives of the network by moving layer by layer from the final to the initial one
- using the chain rule, the derivatives of each layer are multiplied down the network (from the final layer to the initial) to compute the derivatives of the initial layers
- when *n* hidden layers use an activation like the sigmoid function, *n* small derivatives are multiplied together

Vanishing Gradients: shallow vs deep networks

- thus, the gradient decreases exponentially as we propagate down to the initial layers
- a small gradient means that weights & biases of initial layers will not be updated effectively during training
- since initial layers are often crucial to recognise the core elements of input data, this can lead to overall network inaccuracy.

For shallow networks, with only a few layers that use these activations, this isn't a big problem. However, when more layers are used, this can cause the gradient to be too small for training to work effectively.

Vanishing Gradients: Solution 1: Use ReLU

Use activation functions which don't 'squash' the input space into a small region.

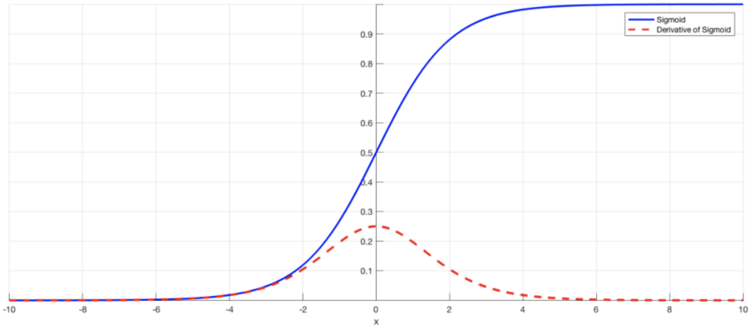
A popular choice is Rectified Linear Unit (ReLU) [8] which maps x to $max(0,x)$.

Vanishing Gradients: Solution 2: Use Batch Normalisation (1)

Problem: when a large input space is mapped to a small one, causing the derivatives to disappear.

sigmoid activation function; $x = wu + b$ for a neuron anywhere in the hidden layers of a NN; u: layer's input; w: weights matrix; b : bias vector

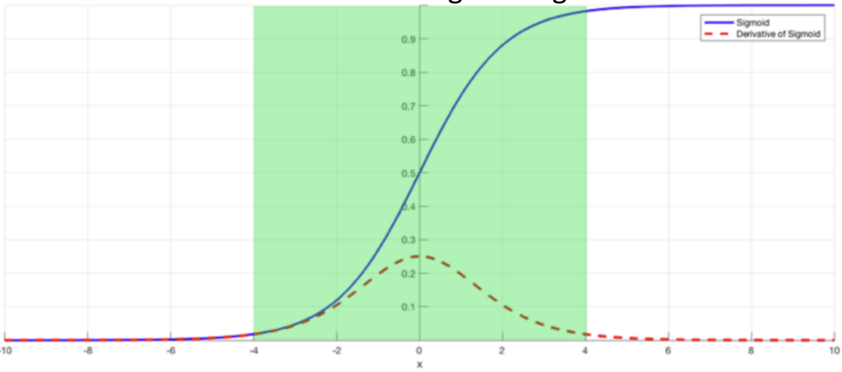
x = very big/small → gradient = 0



Vanishing Gradients: Solution 2: Use Batch Normalisation (2)

Batch Normalisation: Step 1: normalise the input by subtracting its mean and dividing by its standard deviation (ensures zero mean and unit variance)

x doesn't reach outer edges of sigmoid



Vanishing Gradients: Solution 2: Use Batch Normalisation (4)

Batch Normalisation [9]: Step 2: the normalized output of Step 1 is multiplied by a “standard deviation” parameter (gamma; γ) and a “mean” parameter (beta; β) is added to the product

- these two parameters are optimised during network training

- Batch Normalisation increases stability of a Neural Network & speeds up training

Vanishing Gradients: Solution 2: Use Batch Normalisation (5)

Algorithm:

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1\ldots m}\}$;
Parameters to be learned: γ, β

Output: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

$$\sigma_{\mathcal{B}}^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m} (x_i - \mu_{\mathcal{B}})^2 \qquad \text{// mini-batch variance}$$

$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \qquad \text{// normalize}$$

$$y_i \leftarrow \gamma\widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$

Algorithm 1: Batch Normalizing Transform, applied to activation x over a mini-batch.

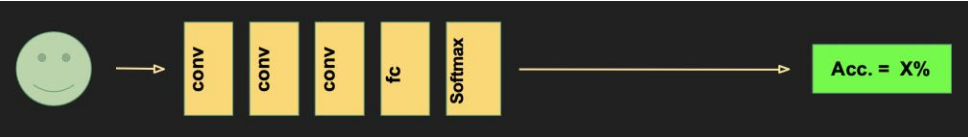
1. Vanishing gradients

2. Degradation problem [10]

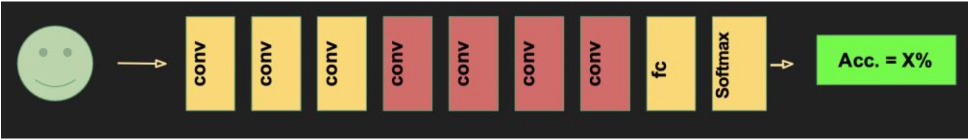
Degradation problem: Definition (1)

Image Classification Problem

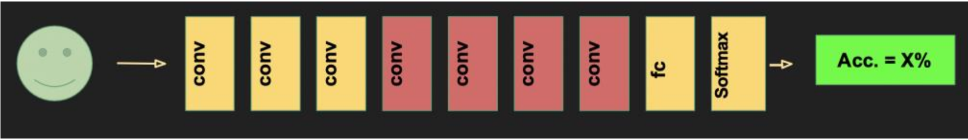
Consider a network having n layers. This network produces some error/accuracy.



Now consider a deeper network with m layers (m>n).



Degradation problem: Definition (2)



When we train this network, we expect it to perform *<u>at least as well as</u>* the shallower network. Why?

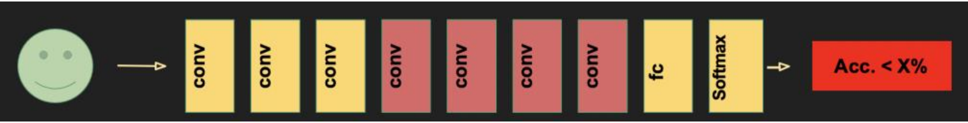
Replace the first n layers of the deep network with the trained n layers of the shallower network. Now replace the remaining $n–m$ layers in the deeper network with an identity mapping (these layers simply output what is fed into them).

Degradation problem: Definition (3)

Thus, our deeper model can easily learn the shallower model's representation.

If there exists a more complex representation of data, we expect the deep model to learn this.

Degradation problem: Definition – In Practice



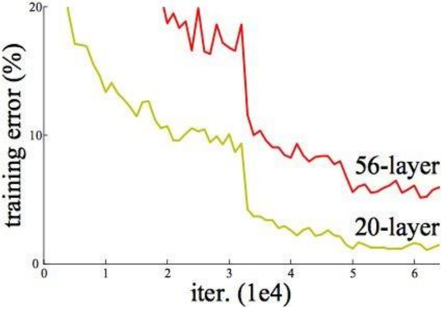
Degradation problem: Definition – Overfitting?

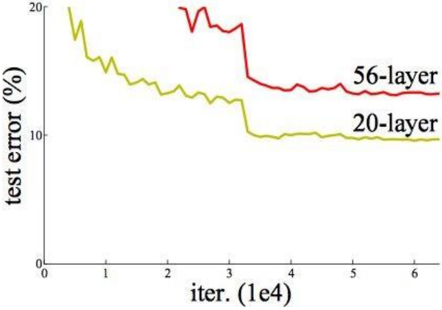
- task is to predict if an image shows a balloon or not
- train a model using a dataset containing many blue colored balloons (and other irrelevant objects)
- test the model on the original dataset: it gives 99% accuracy!
- test the model on a new ("unseen") dataset containing yellow colored balloons: it gives 20% accuracy!

Our model doesn't *generalise* well from our training data to unseen data. This is known as overfitting.

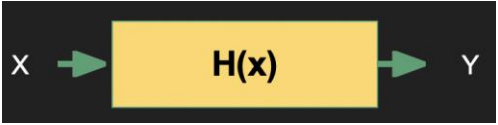
A model that has learned the noise instead of the signal is considered "overfit" because it fits the training dataset but has poor fit with new datasets.

Degradation problem: Definition – Overfitting? No

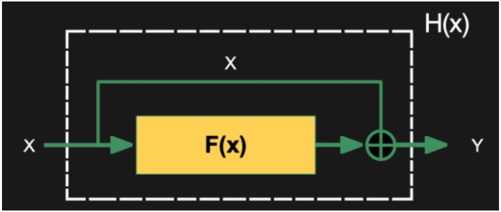




Residual Learning



H(x) is the true mapping function we want to learn



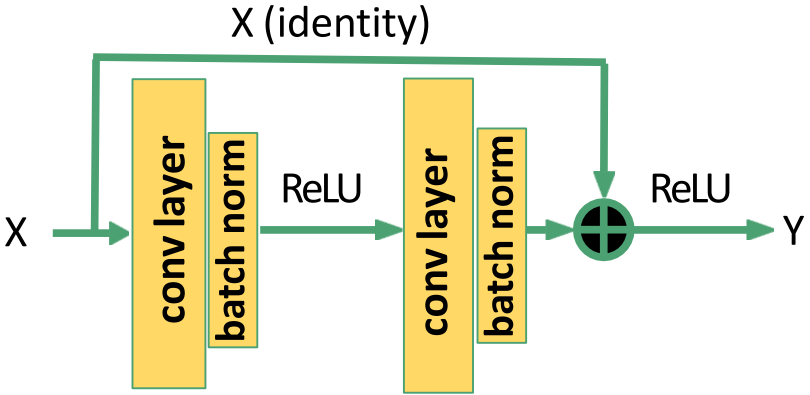
New representation F(x)

$$\boxed{F(x) := H(x) - x}$$

Residual Learning [10]

If F(x) = 0 → identity mapping; if that is a solution the network will be able to find it

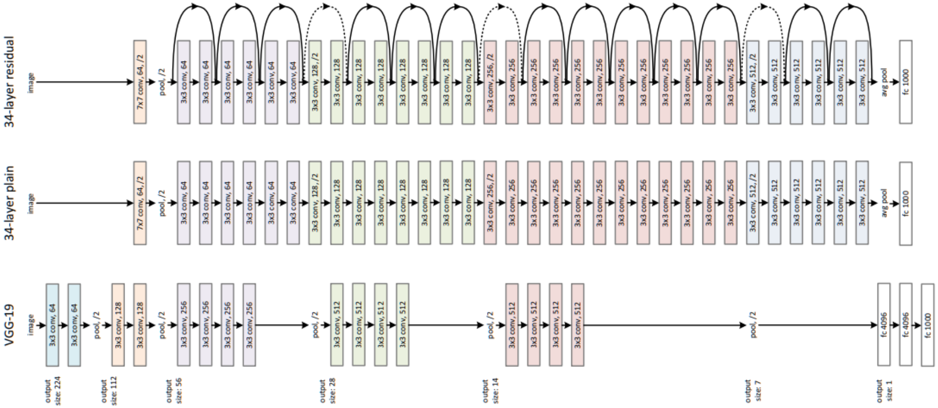
Residual Block (1)



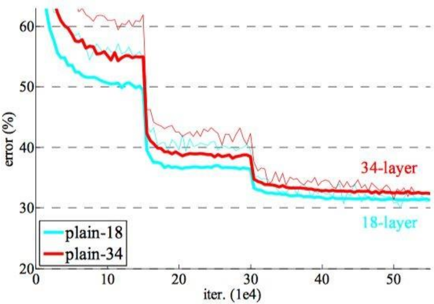
Residual Block (2)

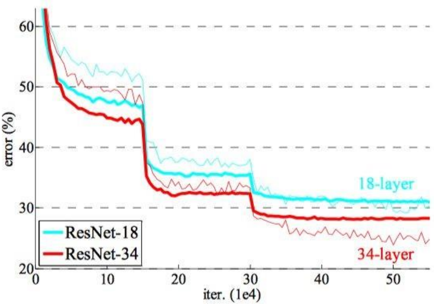
- with this approach, the network will decide how deep it needs to be
- the identity connections introduce no new parameter to the network architecture, hence it will not add any computational burden
- this method allows us to design deeper networks in order to deal with much complicated problems and tasks

ResNet-34: Plain vs With Skip Connections vs VGG (4)



No Degradation for ResNet on ImageNet





Results on ImageNet

method	top-5 err. (**test**)
VGG [40] (ILSVRC'14)	7.32
ResNet (ILSVRC'15)	**3.57**

1. J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li and L. Fei-Fei. **ImageNet: A Large-Scale Hierarchical Image Database.** *IEEE Computer Vision and Pattern Recognition (CVPR), 2009*

2. O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg and L. Fei-Fei. **ImageNet Large Scale Visual Recognition Challenge**. *International Journal of Computer Vision (IJCV),* 2015

3. A. Krizhevsky, I. Sutskever, and G. E Hinton. **Imagenet classification with deep convolutional neural networks**. *Advances in neural information processing systems (NIPS)*, 2012

4. K. Simonyan and A. Zisserman. **Very deep convolutional networks for large-scale image recognition.** *International Conference on Learning Representations (ICLR)*, 2015.

5. https://www.deeplearningbook.org/contents/mlp.html#pf25

6. https://en.wikipedia.org/wiki/Sigmoid_function#cite_note-:0-1

7. https://en.wikipedia.org/wiki/Activation_function

8. https://en.wikipedia.org/wiki/Rectifier_(neural_networks)

9. S. Ioffe and C. Szegedy. **Batch normalization: Accelerating deep network training by reducing internal covariate shift**. *International Conference on Machine Learning (ICML)*, 2015.

10. K. He, X. Zhang, S. Ren, and J. Sun. **Deep residual learning for image recognition.** *IEEE Computer Vision and Pattern Recognition (CVPR), 2016.*