Chapter 11: Residual Networks

Problem 11.1

Derive equation 11.5 from the network definition in equation 11.4.

$$\begin{aligned}\mathbf{h}_1 &= \mathbf{x} + \mathbf{f}_1[\mathbf{x}, \boldsymbol{\phi}_1] \\ \mathbf{h}_2 &= \mathbf{h}_1 + \mathbf{f}_2[\mathbf{h}_1, \boldsymbol{\phi}_2] \\ \mathbf{h}_3 &= \mathbf{h}_2 + \mathbf{f}_3[\mathbf{h}_2, \boldsymbol{\phi}_3] \\ \mathbf{y} &= \mathbf{h}_3 + \mathbf{f}_4[\mathbf{h}_3, \boldsymbol{\phi}_4] \end{aligned} \tag{11.4}$$

$$\begin{aligned}\mathbf{y} = \mathbf{x} &+ \mathbf{f}_1[\mathbf{x}] \\ &+ \mathbf{f}_2[\mathbf{x} + \mathbf{f}_1[\mathbf{x}]] \\ &+ \mathbf{f}_3[\mathbf{x} + \mathbf{f}_1[\mathbf{x}] + \mathbf{f}_2[\mathbf{x} + \mathbf{f}_1[\mathbf{x}]]] \\ &+ \mathbf{f}_4[\mathbf{x} + \mathbf{f}_1[\mathbf{x}] + \mathbf{f}_2[\mathbf{x} + \mathbf{f}_1[\mathbf{x}]] + \mathbf{f}_3[\mathbf{x} + \mathbf{f}_1[\mathbf{x}] + \mathbf{f}_2[\mathbf{x} + \mathbf{f}_1[\mathbf{x}]]]] \end{aligned} \tag{11.5}$$

Substituting $\mathbf{h}_3$ into $\mathbf{y}$, we get:

$$\mathbf{y} = \mathbf{h}_2 + \mathbf{f}_3[\mathbf{h}_2, \boldsymbol{\phi}_3] + \mathbf{f}_4[\mathbf{h}_2 + \mathbf{f}_3[\mathbf{h}_2, \boldsymbol{\phi}_3], \boldsymbol{\phi}_4] \tag{1}$$

Next, substitute $\mathbf{h}_2$ into Equation (1):

$$\mathbf{y} = \mathbf{h}_1 + \mathbf{f}_2[\mathbf{h}_1, \boldsymbol{\phi}_2] + \mathbf{f}_3[\mathbf{h}_1 + \mathbf{f}_2[\mathbf{h}_1, \boldsymbol{\phi}_2], \boldsymbol{\phi}_3] + \mathbf{f}_4[\mathbf{h}_1 + \mathbf{f}_2[\mathbf{h}_1, \boldsymbol{\phi}_2] + \mathbf{f}_3[\mathbf{h}_1 + \mathbf{f}_2[\mathbf{h}_1, \boldsymbol{\phi}_2], \boldsymbol{\phi}_3], \boldsymbol{\phi}_4] \tag{2}$$

Finally, substitute $\mathbf{h}_1$ into the above equation:

$$\begin{aligned}\mathbf{y} = \mathbf{x} &+ \mathbf{f}_1[\mathbf{x}, \boldsymbol{\phi}_1] \\ &+ \mathbf{f}_2[\mathbf{x} + \mathbf{f}_1[\mathbf{x}, \boldsymbol{\phi}_1], \boldsymbol{\phi}_2] \\ &+ \mathbf{f}_3[\mathbf{x} + \mathbf{f}_1[\mathbf{x}, \boldsymbol{\phi}_1] + \mathbf{f}_2[\mathbf{x} + \mathbf{f}_1[\mathbf{x}, \boldsymbol{\phi}_1], \boldsymbol{\phi}_2], \boldsymbol{\phi}_3] \\ &+ \mathbf{f}_4[\mathbf{x} + \mathbf{f}_1[\mathbf{x}, \boldsymbol{\phi}_1] + \mathbf{f}_2[\mathbf{x} + \mathbf{f}_1[\mathbf{x}, \boldsymbol{\phi}_1], \boldsymbol{\phi}_2] + \mathbf{f}_3[\mathbf{x} + \mathbf{f}_1[\mathbf{x}, \boldsymbol{\phi}_1] + \mathbf{f}_2[\mathbf{x} + \mathbf{f}_1[\mathbf{x}, \boldsymbol{\phi}_1], \boldsymbol{\phi}_2], \boldsymbol{\phi}_3], \boldsymbol{\phi}_4] \end{aligned} \tag{3}$$

Equation (3) is equivalent (without the ϕ parameters shown) to Equation (11.5) as required.

Problem 11.2

Unraveling the four-block network in figure 1a produces one path of length zero, four paths of length one, six paths of length two, four paths of length three, and one path of length four. How many paths of each length would there be if with (i) three residual blocks and (ii) five residual blocks? Deduce the rule for K residual blocks

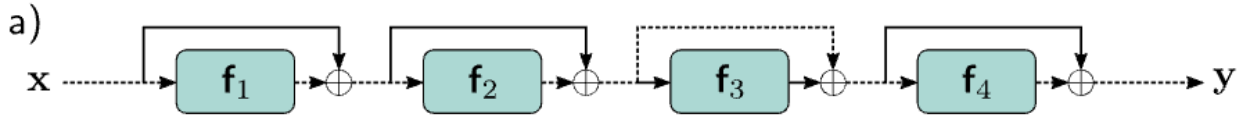

Figure 1: Four-block network

The path of length zero stems from the input $\mathbf{x}$ directly connecting to output $\mathbf{y}$ without passing through any residual blocks. The number of paths of length zero is always one.

In summary, for a network with 4 residual blocks, the number of paths of each length is as follows:

- Paths of length zero: 1
- Paths of length one: 4
- Paths of length two: 6
- Paths of length three: 4
- Paths of length four: 1
- Total paths: 16

For a network with 3 residual blocks, the number of paths of each length is as follows:

- Paths of length zero: 1
- Paths of length one: 3
- Paths of length two: 3
- Paths of length three: 1
- Total paths: 8

For a network with 5 residual blocks, the number of paths of each length is as follows:

- Paths of length zero: 1

- Paths of length one: 5
- Paths of length two: 10
- Paths of length three: 10
- Paths of length four: 5
- Paths of length five: 1
- Total paths: 32

For K residual blocks, the number of paths for each length is given by $\binom{K}{l}$ where K is the number of residual blocks, and l is the length of the path.

$$\binom{K}{l} = \frac{K!}{l!(K-l)!}$$

Problem 11.3

Show that the derivative of the network in equation 11.5 with respect to the first layer $\mathbf{f}_1[\mathbf{x}]$ is given by equation 11.6.

$$\frac{\partial \mathbf{y}}{\partial \mathbf{f}_1} = \mathbf{I} + \frac{\partial \mathbf{f}_2}{\partial \mathbf{f}_1} + \left(\frac{\partial \mathbf{f}_3}{\partial \mathbf{f}_1} + \frac{\partial \mathbf{f}_3}{\partial \mathbf{f}_2}\frac{\partial \mathbf{f}_2}{\partial \mathbf{f}_1}\right) + \left(\frac{\partial \mathbf{f}_4}{\partial \mathbf{f}_1} + \frac{\partial \mathbf{f}_4}{\partial \mathbf{f}_2}\frac{\partial \mathbf{f}_2}{\partial \mathbf{f}_1} + \frac{\partial \mathbf{f}_4}{\partial \mathbf{f}_3}\frac{\partial \mathbf{f}_3}{\partial \mathbf{f}_1} + \frac{\partial \mathbf{f}_4}{\partial \mathbf{f}_3}\frac{\partial \mathbf{f}_3}{\partial \mathbf{f}_2}\frac{\partial \mathbf{f}_2}{\partial \mathbf{f}_1}\right) \tag{11.6}$$

$$\begin{aligned}\mathbf{y} = \mathbf{x} &+ \mathbf{f}_1[\mathbf{x}] \\ &+ \mathbf{f}_2[\mathbf{x} + \mathbf{f}_1[\mathbf{x}]] \\ &+ \mathbf{f}_3[\mathbf{x} + \mathbf{f}_1[\mathbf{x}] + \mathbf{f}_2[\mathbf{x} + \mathbf{f}_1[\mathbf{x}]]] \\ &+ \mathbf{f}_4[\mathbf{x} + \mathbf{f}_1[\mathbf{x}] + \mathbf{f}_2[\mathbf{x} + \mathbf{f}_1[\mathbf{x}]] + \mathbf{f}_3[\mathbf{x} + \mathbf{f}_1[\mathbf{x}] + \mathbf{f}_2[\mathbf{x} + \mathbf{f}_1[\mathbf{x}]]]]\end{aligned} \tag{11.5}$$

First layer contribution:

$$\frac{\partial \mathbf{y}}{\partial \mathbf{f}_1} = \mathbf{I},$$

because the direct contribution of $\mathbf{f}_1$ to $\mathbf{y}$ is linear and independent. The second layer contribution is given by:

$$\frac{\partial \mathbf{y}}{\partial \mathbf{f}_1} \text{ from } \mathbf{f}_2 = \frac{\partial \mathbf{f}_2}{\partial (\mathbf{x} + \mathbf{f}_1)} \cdot \frac{\partial (\mathbf{x} + \mathbf{f}_1)}{\partial \mathbf{f}_1} = \frac{\partial \mathbf{f}_2}{\partial \mathbf{f}_1}$$

Rewriting the third layer as:

$$\mathbf{z}_3 = \mathbf{x} + \mathbf{f}_1[\mathbf{x}] + \mathbf{f}_2[\mathbf{z}_2]$$

Therefore, the direct dependency of $\mathbf{f}_3$ on $\mathbf{f}_1$ is given by:

$$\frac{\partial \mathbf{f}_3}{\partial \mathbf{z}_3} \cdot \frac{\partial \mathbf{z}_3}{\partial \mathbf{f}_1} = \frac{\partial \mathbf{f}_3}{\partial \mathbf{f}_1}$$

The indirect dependency of $\mathbf{f}_3$ on $\mathbf{f}_1$ is given by:

$$\frac{\partial \mathbf{f}_3}{\partial \mathbf{z}_3} \cdot \frac{\partial \mathbf{z}_3}{\partial \mathbf{f}_2} \cdot \frac{\partial \mathbf{f}_2}{\partial \mathbf{f}_1} = \frac{\partial \mathbf{f}_3}{\partial \mathbf{f}_2} \cdot \frac{\partial \mathbf{f}_2}{\partial \mathbf{f}_1}$$

Adding these two terms together gives the third layer contribution to the derivative of $\mathbf{y}$ with respect to $\mathbf{f}_1$:

$$\frac{\partial \mathbf{f}_3}{\partial \mathbf{f}_1} + \frac{\partial \mathbf{f}_3}{\partial \mathbf{f}_2} \cdot \frac{\partial \mathbf{f}_2}{\partial \mathbf{f}_1}$$

A similar method can be followed for the fourth layer to get the derivative of $\mathbf{y}$ with respect to $\mathbf{f}_1$ as given in Equation (11.6).

Problem 11.4

> Explain why the values in the two branches of the residual blocks in figure 11.6a are uncorrelated. Show that the variance of the sum of uncorrelated variables is the sum of their individual variances.

The values in the two branches of the residual blocks in figure 11.6a are uncorrelated beacause the input to the main branches of the residual block is multiplied by the weight matrix, which is normally distributed with mean zero ($\Omega \sim \mathcal{N}(0, \Sigma)$). This can equally multiply by a positive or negative number, so they are uncorrelated.

The definition of variance is given by:

$$\text{Var}(X) = \mathbb{E}[X^2] - \mathbb{E}[X]^2$$

For two uncorrelated variables X and Y, we want to show that the variance of their sum is the sum of their individual variances:

$$\begin{aligned}
\text{Var}(X+Y) &= \mathbb{E}[(X+Y)^2] - \mathbb{E}[(X+Y)]^2 \\
&= \mathbb{E}[X^2 + 2XY + Y^2] - (\mathbb{E}[X] + \mathbb{E}[Y])^2 \\
&= \mathbb{E}[X^2] + 2\mathbb{E}[XY] + \mathbb{E}[Y^2] - (\mathbb{E}[X] + \mathbb{E}[Y])^2 \\
&= \mathbb{E}[X^2] - \mathbb{E}[X]^2 + 2\mathbb{E}[XY] - 2\mathbb{E}[X]\mathbb{E}[Y] + \mathbb{E}[Y^2] - \mathbb{E}[Y]^2 \\
&= \mathbb{E}[X^2] - \mathbb{E}[X]^2 + \mathbb{E}[Y^2] - \mathbb{E}[Y]^2 \\
&= \text{Var}(X) + \text{Var}(Y)
\end{aligned}$$

Where we used the fact that X and Y are uncorrelated so have zero covariance to go from line 4 to line 5:

$$\text{Cov}(X,Y) = \mathbb{E}[XY] - \mathbb{E}[X]\mathbb{E}[Y] = 0 \implies \mathbb{E}[XY] = \mathbb{E}[X]\mathbb{E}[Y]$$

Problem 11.5

The forward pass for batch normalisation given a batch of scalar values $\{z_i\}_{i=1}^I$ consists of the following operations:

$$\begin{aligned}
f_1 &= \mathbb{E}[z_i] & f_5 &= \sqrt{f_4 + \epsilon} \\
f_{2i} &= x_i - f_1 & f_6 &= \frac{1}{f_5} \\
f_{3i} &= f_{2i}^2 & f_{7i} &= f_{2i} \times f_6 \\
f_4 &= \mathbb{E}[f_{3i}] & z_i' &= f_{7i} \times \gamma + \delta
\end{aligned} \tag{11.10}$$

where $\mathbb{E}[z_i] = \frac{1}{I}\sum_i z_i$. Write Python code to implement the forward pass. Now derive the algorithm for the backward pass. Work backward through the computational graph computing the derivatives to generate a set of operations that computes $\frac{\partial z_i'}{\partial z_i}$ for evert element in batch. Write Python code to implement the backward pass.

Coding questions not included.

Problem 11.6

Consider a fully connected neural network with one input, one output, and ten hidden layers, each of which contains twenty hidden units. How many parameters does this network have? How many parameters will it have if we place a batch normalisation operation between each linear transformation and ReLU?

For a fully connected neural network with one output, and ten hidden layers, each of which contains twenty hidden units, there are $3(20) + 1 + (10 - 1) \times 20 \times (20 + 1) = 3841$ parameters. (see problem 4.10)

Placing a batch normalisation operation between each linear transformation and ReLU will add $2\times$ number of features per layer, i.e., $2 \times 20 = 40$ parameters per hidden layer and $2 \times 1 = 2$ parameters for the output layer. This results in an additional $2 \times 20 \times 10 = 400$ parameters for the hidden layers and 2 parameters for the output layer. Therefore, the total number of parameters will be $3841 + 400 + 2 = 4243$.

Problem 11.7

Consider applying an L2 regularization penalty to the weights in the convolutional layers in figure 11.7a, but not to the scaling parameters of the subsequent BatchNorm layers. What do you expect will happen as training proceeds?

The penalty will prefer small values for the weights, and this will be compensated for by the BatchNorm scaling. Eventually, the weights will become very small, and this may cause numerical problems.

Problem 11.8

Consider a convolutional residual block that contains a batch normalisation operation, followed by a ReLU activation function, and then a 3×3 convolutional layer. If the input and output both have 512 channels, how many parameters are needed to define this block?

Now consider a bottleneck residual block that contains three batch normalization/ReLU/-convolution sequences. The first uses a 1×1 convolution to reduce the number of channels from 512 to 128. The second uses a 3×3 convolution with the same number of input and output channels. The third uses a 1×1 convolution to increase the number of channels from 128 to 512 (see figure 11.7b). How many parameters are needed to define this block?

For the convolutional residual block with 512 input and output channels, batch normalisation has two parameters per channel, and the convolutional layer has $3 \times 3 \times 512 \times 512 = 2,359,296$ parameters. Therefore, the total number of parameters is $2 \times 512 + 2359296 = 2,360,320$.

For the bottleneck residual block, the first layer will have $2 \times 512 = 1024$ parameters from the batch normalisation and $1 \times 1 \times 512 \times 128 = 65,536$ weights from the 1×1 convolution. The second layer will have $2 \times 128 = 256$ parameters from the batch normalisation and $3 \times 3 \times 128 \times 128 = 147,456$ weights from the 3×3 convolution. The third layer will have $2 \times 128 = 256$ parameters from the batch normalisation and $1 \times 1 \times 128 \times 512 = 65,536$ weights from the 1×1 convolution. Therefore, the total number of parameters is $1024 + 65536 + 256 + 147456 + 256 + 65536 = 280,064$. This design effectively reduces the computational cost compared to a block with larger convolutions directly applied to high-dimensional inputs.

Problem 11.9

The U-Net is completely convolutional and can be run with any sized image after training. Why do we not train with a collection of arbitrarily-sized images?

Fixed-size inputs ensure that the model learns features that are spatially standardized across different training examples. Arbitrarily sized images could lead to features that are not comparable, as the receptive field of the convolutional filters might cover vastly different regions of interest across different images. Additionally, training with arbitrarily-sized images is not practical due to computational, performance, and memory constraints