2.2. Machine learning models

Throughout the project, various machine-learning algorithms will be used to train and evaluate the effectiveness of using contrastive learning. This section provides a summary of these algorithms.

2.2.1. Logistic regression

Logistic regression is an established statistical model that is used for classification problems. Logistic regression estimates the ~~target values~~ probability $\hat{\boldsymbol{y}}$ of a binary target value by applying the sigmoid function $\sigma(\cdot)$ to a linear transformation of the input data:

$$\hat{y} = \sigma(z) \tag{2.1}$$

where z is calculated as:

$$z = \sum_i \boldsymbol{x_i} \times \boldsymbol{w_i} + b \tag{2.2}$$

and the sigmoid function is defined as

$$\sigma(z) = \frac{1}{1 + e^{-z}} \tag{2.3}$$

and maps values between $(-\infty, \infty)$ to a value between $(0, 1)$. The model takes in the input data x_i and multiplies each component of the input vector with a weight w_i, where the weights are optimised during training to maximise the likelihood criterion. The result is passed to the sigmoid/standard logistic function which maps it to between 0 and 1. The output can be interpreted as the probability that the particular input variable belongs to the positive class, i.e. $\hat{y} = P(y = 1|\boldsymbol{x}, \boldsymbol{w})$ where $y = 1$ if it belongs to the specified class, and 0 otherwise. Logistic regression can also be seen as a single node of a multi-layer perceptron with a sigmoid activation function as shown in Figure 2.1.

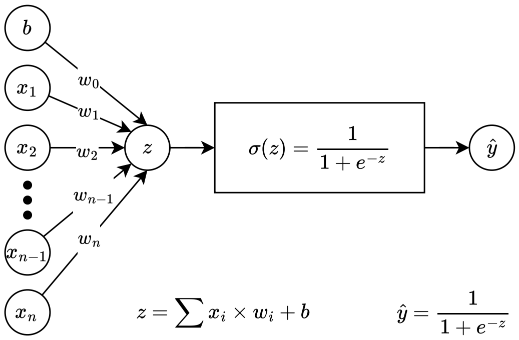

b
x_1
x_2
x_{n-1}
x_n
w_0
w_1
w_2
w_{n-1}
w_n
z
$\sigma(z) = \frac{1}{1+e^{-z}}$
$\hat{y}$
$z = \sum x_i \times w_i + b$
$\hat{y} = \frac{1}{1+e^{-z}}$

Figure 2.1: Logistic regression can be viewed as a single node of a multi-layer perception with input $\boldsymbol{x}$, a sigmoid activation function $\sigma(\cdot)$ and output $\hat{y}$.

2.2.2. Multi-layer perceptron

A neuron is a biological structure in the brain that processes and transmits nerve signals. The artificial neuron, also referred to as a perceptron, is based on the biological neuron as shown in Figure 2.2. It receives input $\boldsymbol{x}$ from a source, adds a level of importance $\boldsymbol{w}$ to each input signal and generates outputs $\boldsymbol{y}$.

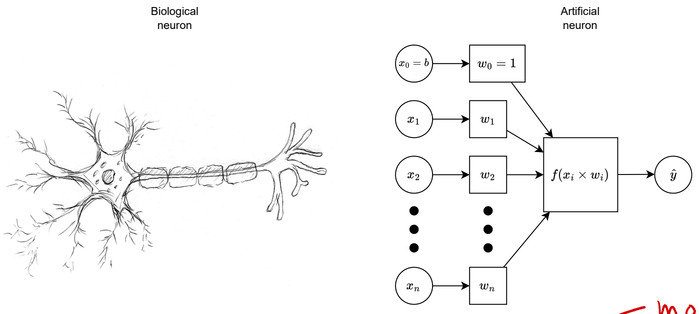
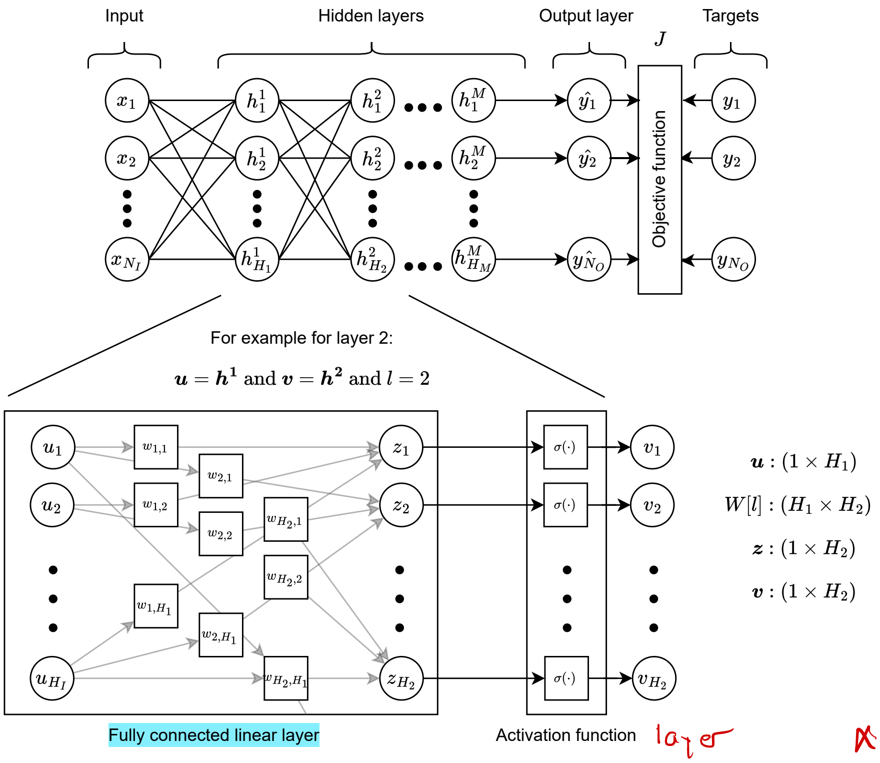


Figure 2.2: Biological and artificial ~~neuron~~. model (right).

neuron its (left)

Sorry could not read this somehow

A multi-layer perceptron (MLP) consists of many of these neurons/perceptrons arranged in layers as shown in Figure 2.3. It generally consists of input $\boldsymbol{x}$, M hidden layers, and the final output $\hat{y}$, where the final output can be either a single value for binary classification or multiple values for multinomial classification.



layer

Figure 2.3: Multi-layer perceptron with input layer x with N_I inputs, M hidden layers each with $H_1, H_2, ..., H_M$ nodes, and outputs $\boldsymbol{\hat{y}}$.

A multi-layer perceptron is trained using a forward pass followed by a backward pass. First, the input data and the weights are initialised. For each layer in the MLP, the input vector $\boldsymbol{u}$ is multiplied by the weights $\boldsymbol{W}$. A non-linear transformation $\sigma(\cdot)$, referred to as the activation function, is applied to the resulting vector $\boldsymbol{z}$ to generate the output vector of each layer $\boldsymbol{v}$. The output of the current layer is then used as input to the next layer and the output $\hat{\boldsymbol{y}}$ of the last layer is the network's output as a whole. The outputs are compared to the true labels or targets using a loss function. For example, if the squared error is used as a loss function, then

$$J = \frac{1}{2}(\boldsymbol{y} - \hat{\boldsymbol{y}})^T(\boldsymbol{y} - \hat{\boldsymbol{y}}) \tag{2.4}$$

To optimise the weights of the multi-layer perceptron, the loss function is minimised to minimise the difference between the ground truth and the output. After ~~the model has passed through the entire~~ a forward pass has been performed, the derivative of the loss function J relative to the weights for each layer $W[l]$ is calculated:

$$\frac{\partial J}{\partial W[l]} = \left[\frac{\partial J}{\partial \boldsymbol{w}_1[l]}, \frac{\partial J}{\partial \boldsymbol{w}_2[l]}, \ldots, \frac{\partial J}{\partial \boldsymbol{w}_{H_l}[l]}\right] \tag{2.5}$$

bold?

for node i in layer l:

$$\frac{\partial J}{\partial \boldsymbol{w}_i[l]} = \frac{\partial \boldsymbol{z}[l]}{\partial \boldsymbol{w}_i[l]} \times \frac{\partial \boldsymbol{v}[l]}{\partial \boldsymbol{z}[l]} \times \frac{\partial J}{\partial \boldsymbol{v}[l]} \tag{2.6}$$

These derivatives are stored and used to update the weights during back-propagation. The pseudo-code for the forward pass is provided in ~~the~~ Appendix, C.1. Commonly used activation functions include linear, sigmoid, ~~softmax~~, tanh and ReLU. A summary of each of these activation functions is provided in ~~the~~ Appendix, A.1.

Softmax is only used at the output of a multinomial classifier so I would not mention it here.

2.2.3. Convolutional neural network

A convolutional neural network (CNN) consists of one or more convolutional layers which each include one or more filters, pooling layers and fully connected layers. It is a deep-learning technique that takes advantage of two-dimensional filtering to perform feature extraction and is particularly effective for ~~on~~ images. A simple convolutional network is shown [can be seen] in Figure 2.4 ~~to show~~ indicating, how an image is [processed,] convolved, pooled and then presented as input to a feedforward network. at the output. ? ?

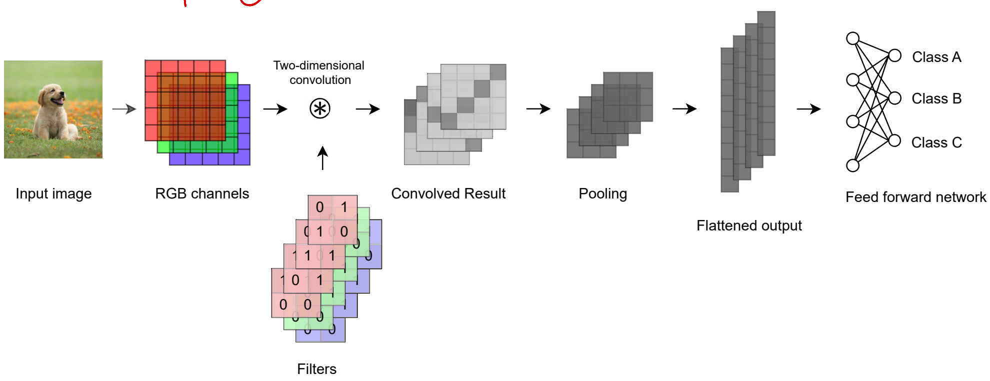
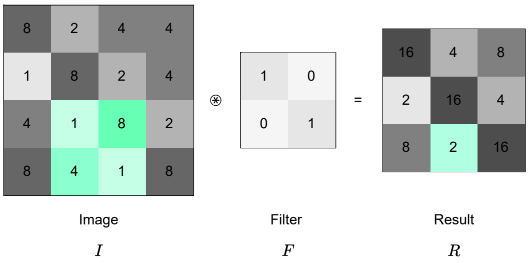


Figure 2.4: A convolutional neural network receives input in the form of an image. The input image is usually composed of three channels: red, green and blue (RGB). These channels are then convolved with a set of filters to perform feature detection. Pooling is used to reduce the dimension of the resulting feature maps. Finally, the output of the convolutional network is rearranged as a set of vectors (flattened) and fed to a fully connected feed-forward network.

A convolutional layer uses one or more filters for feature detection. Figure 2.5 shows how an input image, represented as a matrix, is convolved with a filter. The filter passes over the entire input matrix, at each offset performing an element-wise multiplication between [the input pixels I and] the filter pixels F, and □ with which they overlap, storing the sum of the result in the corresponding element ~~in~~ of the feature map R.



8	2	4	4
1	8	2	4
4	1	8	2
8	4	1	8

⊛

1	0
0	1

=

16	4	8
2	16	4
8	2	16

$$R[n_i, n_j] = \sum_{i=n_i}^{n_i+F_{height}} \sum_{j=n_j}^{n_j+F_{width}} I[i,j]F[i-n_i, j-n_j]$$

Figure 2.5: An illustrative example of Convolution ~~explained~~ using a diagonal line as a filter. As The filter moves over the entire input image, each element in the input image is multiplied element-wise with the corresponding element in the feature map, and the total sum is stored [as the output layer.] at the corresponding location in the result image R.

Mathematically, convolution in a CNN is implemented as cross-correlation:

$$r_{I,F}[n_i, n_j] = \sum_{i=n_i}^{n_i+F_{height}} \sum_{j=n_j}^{n_j+F_{width}} I[i,j] \times F[i-n_i, j-n_j] \tag{2.7}$$

where a filter image F is passed over an image I to produce a smaller image with dimensions $(I_{width} - F_{width} + 1, I_{height} - F_{height} + 1)$. For example, if $n_i = 3, n_j = 2, F_{height} = 2, F_{width} = 2$ as shown in turquoise in the figure, then:

$$\begin{aligned} r_{I,F}[3,2] &= \textstyle\sum_{i=3}^{5} \sum_{j=2}^{4} I[i,j]F[i-3, j-2] \\ r_{I,F}[3,2] &= I[3,2]F[0,0] + I[3,3]F[0,1] + I[4,2]F[1,0] + I[4,3]F[1,1] \\ r_{I,F}[3,2] &= 1*1 + 8*0 + 4*0 + 1*1 \\ r_{I,F}[3,2] &= 2 \end{aligned} \tag{2.8}$$

This was Fig 2.4?

Figure 2.6 shows a convolutional layer in more detail. It consists of an input image I with the respective RGB channels. The RGB channels are convolved with the RGB channels of each filter F, to form convolved images R. The convolved images are then passed to an activation function similar to MLPs, resulting in an output matrix V. This entire process is referred to as a convolutional layer.

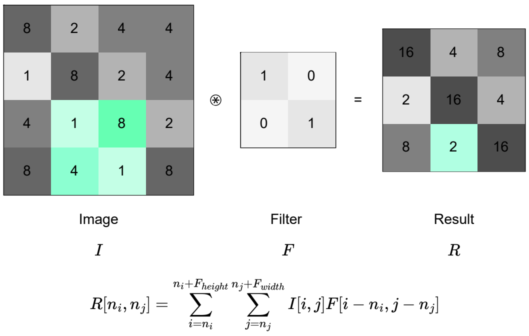

8 2 4 4
1 8 2 4
4 1 8 2
8 4 1 8
⊛
1 0
0 1
=
16 4 8
2 16 4
8 2 16
Image
I
Filter
F
Result
R
$R[n_i,n_j] = \sum_{i=n_i}^{n_i+F_{height}} \sum_{j=n_j}^{n_j+F_{width}} I[i,j]F[i-n_i, j-n_j]$

Figure 2.6

Pooling layers are used to reduce the dimension of the feature maps. Similar to ~~the~~ a convolutional layer, pooling takes a mask with fixed dimensions and aggregates the image values that fall within the mask. Max-pooling obtains the maximum feature map value in the feature map, while average-pooling obtains the average. Finally The fully connected layers are as described in the previous section, where the output of ~~the~~ a convolutional ~~layers~~ and pooling layer~~s~~ is ~~are~~ used as input to the fully connected layer~~s~~.

within the mask

or ?

2.2.4. ResNet

A residual network (ResNet) is a convolutional neural network that aims to solve the problem of vanishing gradients which plagues deep architectures by introducing Residual Blocks. A Residual Block consists of two convolutional layers with a skip connection as shown in Figure 2.7. The skip connection allows the model to bypass convolutional layers during training. This allows for deeper models to be trained more successfully.

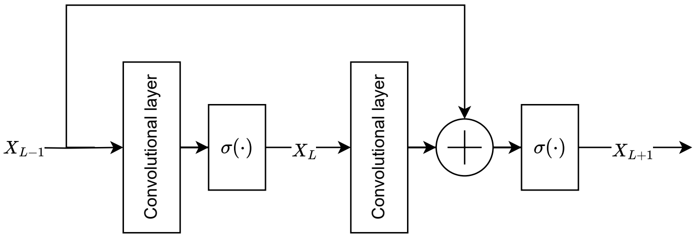


Figure 2.7: A resnet convolutional block showing the skip connection.

A ResNet consists of several of these residual blocks in series. Figure 2.8 shows the general layout of a ResNet18 model (18 refers to the total number of connected layers). The model starts with a 7 × 7 convolutional layer followed by a max pooling layer. It then consists of four ~~ResNet~~ RN convolutional blocks that are each repeated twice (for different ResNet architectures, the number of such repeated blocks differs). ~~where~~ Each convolutional block consists of two convolutional layers with ~~the~~ a skip implemented as described above. The model ends with an average pooling layer and a fully connected layer.

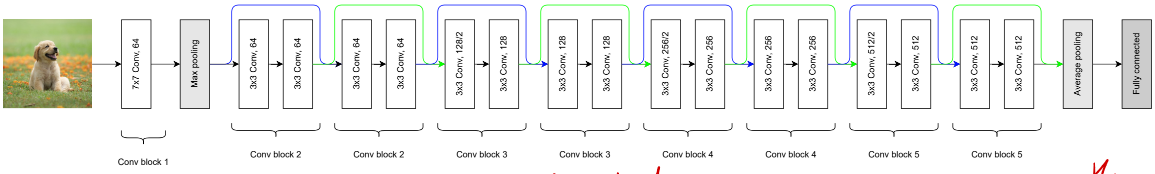


Figure 2.8: Structure of a ResNet18 model, ~~consisting of a~~ showing the initial convolutional ~~layer, a~~ and max pooling layer, the four convolutional blocks (repeated twice), ~~an~~ the subsequent average pooling layer and ~~a~~ the final fully connected layer.