Chapter 2

Background information

This chapter will discuss and explain some of the concepts that are used in this project. This includes a list of definitions, a brief discussion of the different machine-learning models used throughout the project, a summary of the mainstream contrastive learning architectures, short explanations of commonly used contrastive loss functions and a description of the different acoustic features that can be used as input to the models.

2.1. Definitions

This section will provide a few definitions of concepts that are used in this field that are not common knowledge or well-defined in the literature.

2.1.1. Self-supervised learning

Supervised learning uses labelled data to train models to perform classification. In contrast, unsupervised learning uses unlabelled data to learn patterns and clusters. Self-supervised learning uses unlabeled data to generate representations that contain inherent characteristics of the data. This is also referred to as pretext learning or predictive learning. The idea behind self-supervised learning is to address the challenge posed by the usually limited amount of labelled data available. The model solves unsupervised problems by implicitly generating representations that can be used as labels that describe the input data in a way that encodes different characteristics of the input data.

2.1.2. Representation Learning

The word representation is used in the literature to refer to the output of an encoder such as a ResNet model. In the context of this project, representation refers to the embedding of input data to a new space (which could be lower or higher dimensional), that encapsulates the characteristics of the data in a format that is more useful for analysis or further processing than the input data itself. Representation learning is the field of machine learning that specialises in building models that are used to generate these representations.

2.1.3. Contrastive learning

Contrastive learning is a self-supervised representation learning technique that aims to learn low-dimensional representations of high-dimensional input data by contrasting the input data such that similar inputs are close to each other in the representation space and dissimilar inputs are further away in the representation space. This can be done by using an architecture known as a Siamese network and a contrastive loss function to minimise the distance between similar images. The definition of a "similar image" depends on the architecture and/or the loss function. In some instances, a similar image refers to any image with the same label and in other instances a similar image refers to an augmented sample of the input image. This also means that, depending on the architecture and/or the loss function, contrastive learning can either be a self-supervised, unsupervised or supervised learning technique.

2.1.4. Augmentations

To contrast the true input, each image is augmented to create a pair consisting of the true image and the augmented image. The augmentation process takes the image and applies one or more image augmentation techniques to each image. The augmentation processes considered include image augmentation processes (such as random blur, Gaussian blur and random crop) and audio augmentation processes (such as time shift, frequency shift and fade in and out). The augmented image is also referred to as the transformed image or the positive sample.

2.1.5. Transfer learning

Transfer learning is a technique where a pre-trained model is used to transfer knowledge to another model. During the initial stages of training low-level features are learned and later on the model is fine-tuned to the specific task. Transfer learning separates the low-level training from the fine-tuning process to maximise. A base model is pre-trained using large datasets that are similar to the specific task at hand. The base model is used to generate features that encode some of the characteristics of the large dataset. The features are then used as input to the task-specific model. This process enables the task-specific model to fine-tune faster and more effectively using the data available.

2.1.6. Deep learning

Deep learning refers to machine-learning techniques that mimic human intelligence. Deep learning models use neural network models to solve complex problems and include multi-layer perceptrons, convolutional neural networks and recurrent neural networks.

2.2. Machine learning models

Throughout the project, different machine-learning algorithms will be used to train and evaluate the effect of using contrastive learning. This section provides a summary of the different machine-learning algorithms that are used throughout the project. A logistic regression model and a multi-layer perceptron model will be used to evaluate the performance of the final model. The final model will be a deep learning model that uses a Siamese network that consists of an encoder model (residual network model) and a projection head (multi-layer perceptron).

2.2.1. Logistic regression

Logistic regression is a well-known statistical model that is used to predict future target values based on historical data. Logistic regression estimates the target values $\hat{\boldsymbol{y}}$ by applying the sigmoid function $\sigma(\cdot)$ to the weighted input data with a constant factor b:

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

where $\boldsymbol{z}$ is calculated as:

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

and the sigmoid function is defined as:

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

and maps values between $(-\infty, \infty)$ to a value between $(0, 1)$. Logistic regression can be seen as a multi-layer perceptron without a hidden layer with a sigmoid activation function as shown in Figure 2.1.

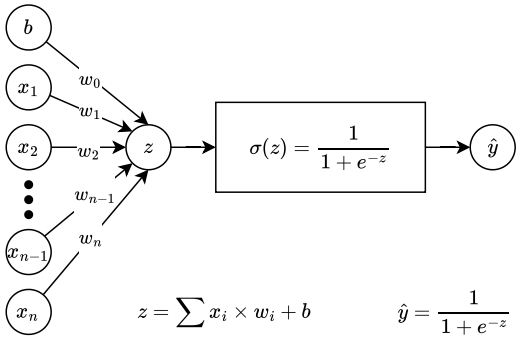

Figure 2.1: Logistic regression is shown as a multi-layer perception with input $\boldsymbol{x}$, a sigmoid activation function $\sigma(\cdot)$ and output $\hat{\boldsymbol{y}}$.

2.2.2. Multi-layer perceptron

A multi-layer perceptron (MLP) consists of an input layer $\boldsymbol{x}$, M hidden layers and an output layer which consists of an activation function $\sigma(\cdot)$ that maps the output from the last hidden layer to the final output $\hat{\boldsymbol{y}}$. Each element of the input layer is connected to each element of the first hidden layer, forming a fully connected layer. Similarly, all the hidden layers are fully connected. The activation function is used to map the output from one domain (output from the last hidden layer) to another domain (output space). Commonly used activation functions include linear, sigmoid, softmax, tanh and ReLU.

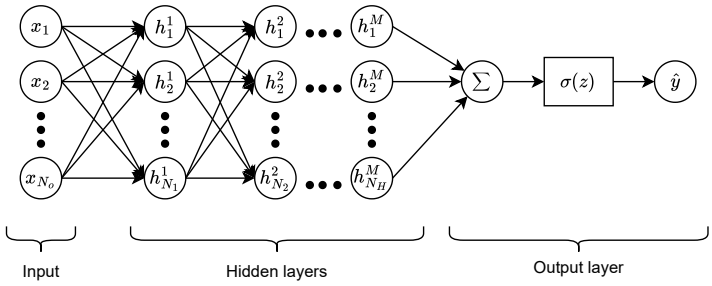

Figure 2.2: Multi-layer perceptron with input layer x with N_O inputs, M hidden layers each with sizes $N_1, N_2, ..., N_H$ respectively, and the output layer that uses an activation function $\sigma(z)$ to generate targets $\hat{\boldsymbol{y}}$.

2.2.3. Convolutional neural network

A convolutional neural network (CNN) consists of convolutional layers, filters, pooling layers and fully connected layers as seen in Figure 2.3. It is deep deep-learning technique that uses filters to perform feature extraction on images.

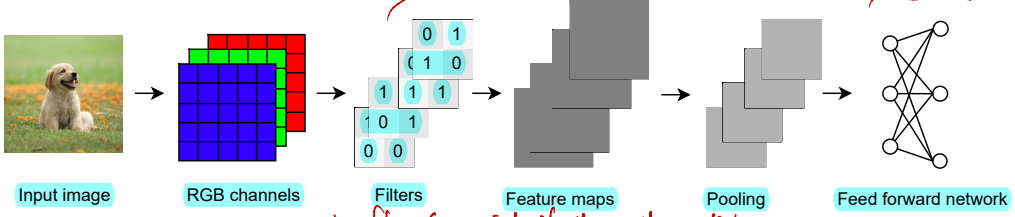

Figure 2.3: A convolutional neural network receives input in the form of images. The input image is split into red, green and blue (RGB) channels. These channels are then convoluted with the filters to perform feature detection. Pooling is used to reduce the dimension of the feature maps. Finally, the output of the convolutional network is fed to a fully connected feed-forward network.

A convolutional layer uses a filter (in this case the filter represents a diagonal line) for feature detection as shown in Figure 2.4. The filter passes over the entire image, performing an element-wise multiplication between the image and the filter elements, storing the total result in the corresponding element in the feature map.

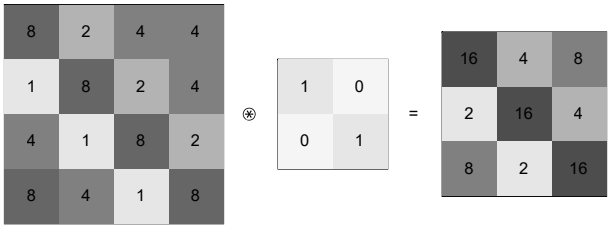

Figure 2.4: Convolution explained using a diagonal line as a filter. The filter moves over the entire input image, each element in the input image is multiplied element-wise with the corresponding element in the feature map, and the total sum is stored as the output layer.

A pooling layer is used to reduce the dimension of the layers. Similar to the convolutional layer, pooling takes a filter (with no weights) and aggregates the result. Max-pooling stores the maximum value in the feature map, and average-pooling stores the average in the feature map. The fully connected layers are as described above, where the output of the convolutional layers and pooling layers are used as input to the fully connected layers.

2.2.4. ResNet

A ResNet aims to solve the problem of vanishing gradients when using a deep (many layers) CNN using Residual Blocks. A Residual Block consists of two convolutional layers with a skip connection as shown in Figure 2.5. Instead of training the convolutional layer, the input X_{L-1} skips the training and gets added to the output of the next convolutional layer X_L. The skip connection allows the model to skip some of the convolutional layers during training, allowing for deeper models without influencing the performance.

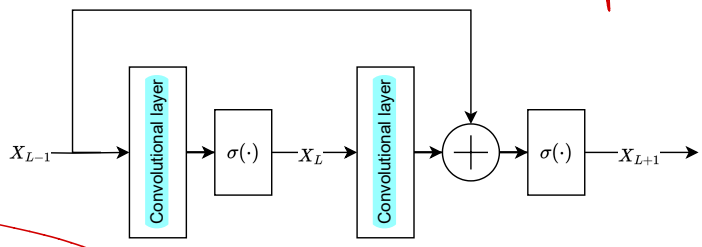

Figure 2.5: Skip Connection used in Residual Networks.

A Residual network consists of multiple of these Residual blocks. Figure 2.6 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 convolutional blocks that are repeated twice (for different ResNet architectures, the number of repeated blocks differs), where each convolutional block consists of two convolutional layers with the skip implemented as described above. The model ends with an average pooling layer and a fully connected layer.

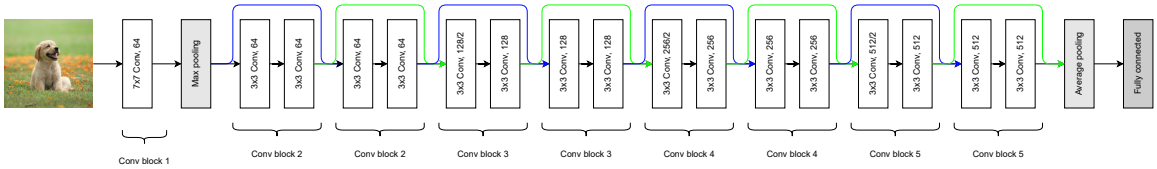

Figure 2.6: ResNet18 model consisting of a convolutional layer, a max pooling layer, four convolutional blocks (repeated twice), an average pooling layer and a fully connected layer.

2.2.5. Siamese network

A Siamese network is a neural network that consists of two sub-networks, where the sub-networks share the same weights. Figure 2.7 shows the layout of a basic Siamese network where the two sub-networks are shown as parallel branches. Typically, the input for this model will be a high dimensional representation, such as a mel spectrogram image, while the features generated will be a low dimensional representation. Each sub-network consists of an encoder $f(\cdot)$, which returns a representation of the input, followed by a projection head $g(\cdot)$ which returns the features $\boldsymbol{z_i}$ and $\boldsymbol{z_j}$. The encoder model typically uses a ResNet model to encode the input images into representations. The projection head uses a multi-layer perceptron to generate features that can then be used as input to another model. A contrastive loss function is used to maximise the agreement between the two input samples.

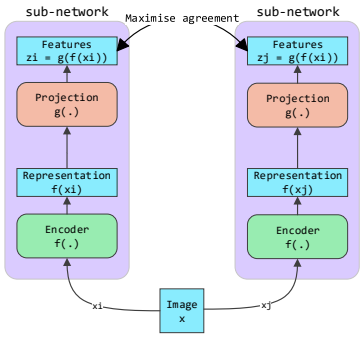

Figure 2.7: A general Siamese network takes in an image $\boldsymbol{x}$, creates a copy of the original input image $\boldsymbol{x_i}$ and an augmentation of the original image $\boldsymbol{x_j}$. Each sub-network consists of an encoder $f(\cdot)$, and a projection head $g(\cdot)$ which generates features $\boldsymbol{z_i}$ and $\boldsymbol{z_j}$. The loss function uses the features generated to maximise the agreement between the two sub-networks.

Gradient collapse: Gradient collapse occurs when the gradients of the loss function used in Siamese networks become very small. This is a common phenomenon in contrastive learning where loss functions are used to minimise the distance between similar images. Each of the architectures discussed below has a unique way of ensuring that the architecture does not collapse.

2.3. Contrastive learning architectures

The following section will give a summary of the mainstream architectures used to perform contrastive learning. Six image-specific contrastive learning architectures will be discussed, as well as two audio-specific architectures.

2.3.1. SimCLR

SimCLR is a self-supervised learning architecture that ~~is~~ specifically ~~designed~~ for image processing and is used to perform contrastive learning of visual representations [8]. Each input image $\boldsymbol{x_i}$ is randomly transformed using one of the augmentation processes to form positive pairs $\boldsymbol{x_i}, \boldsymbol{x_j}$, and negative pairs $\boldsymbol{x_i}, \boldsymbol{x_{k \neq i,j}}$. The augmentation processes consist of random cropping, random colour distortions, and random Gaussian blur. A base encoder $f(\cdot)$ (usually a ResNet) extracts representation vectors from the augmented pairs. A projection head $g(\cdot)$ consisting of a linear dense layer, followed by a ReLU activation function and another linear dense layer, maps the representation vectors into a space where the loss function can be applied. Lastly, a contrastive loss function $\mathcal{L}_{i,j}$ is used to minimise the distance between similar images and maximise the distances between dissimilar images. Figure 2.8 shows the SimCLR architecture.

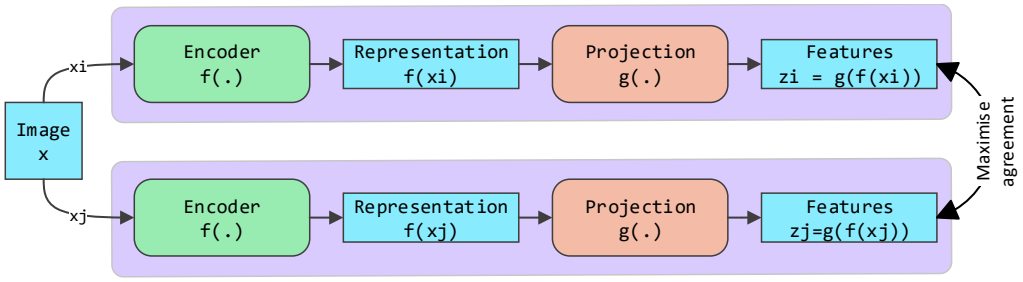

Figure 2.8: The SimCLR architecture takes an input image $\boldsymbol{x}$ and creates two instances, the anchor $\boldsymbol{x_i}$ and the augmentation $\boldsymbol{x_j}$. An encoder $f(\cdot)$ is used to create representation $f(\boldsymbol{x_i})$ and $f(\boldsymbol{x_j})$. These representations then pass through a projection head $g(\cdot)$ to form features $\boldsymbol{z_i}$ and $\boldsymbol{z_j}$. A contrastive loss function is used to minimise the distance between similar images and maximise the distance between dissimilar images.

2.3.2. MoCo

Momentum Contrast (MoCo) is an unsupervised representation learning architecture [9]. It builds dictionaries that are large and consistent, where a dictionary is a queue of encoded representations of the data samples. The architecture receives input in the form of an image which is referred to as the query. An augmented image is created from the query which is referred to as the key. MoCo uses two encoders, a query encoder and a