Pattern recognition of movement through Machine Learning – utilisation of Machine Learning to identify defined ballet dance positions in a ballet movement dictionary.

Gráinne Mulvey

Supervisor: Dr Greg Doyle
Carlow Institute of Technology

A thesis submitted to Carlow Institute of Technology in partial fulfilment of the requirements for the diploma of Master of Science in Data Science

2018

Work submitted for assessment which does not include this declaration will not be assessed.

INSTITUTE of TECHNOLOGY CARLOW

DECLARATION

*I declare that all material in this submission e.g. thesis/essay/project/assignment is entirely my own work except where duly acknowledged.

*I have cited the sources of all quotations, paraphrases, summaries of information, tables, diagrams or other material; including software and other electronic media in which intellectual property rights may reside.

*I have provided a complete bibliography of all works and sources used in the preparation of this submission.

*I understand that failure to comply with the Institute's regulations governing plagiarism constitutes a serious offence.

Student Name: (Printed) _____

Student Number: _____

Signature: _____

Date: _____

Please note:

a) * Individual declaration is required by each student for joint projects.

b) Where projects are submitted electronically, students are required to type their name under signature.

c) The Institute regulations on plagiarism are set out in Section 10 of Examination and Assessment Regulations published each year in the Student Handbook.

ABSTRACT

Convolutional Neural Networks (CNN) are a key factor in making self-driving cars become a reality. The possibility of using a CNN to identify movement, particularly specific movement, seems an achievable task. However, identifying the correct CNN for the task is not without difficulty. CNNs are the Neural Networks used for pattern recognition. Creating a limited number of ballet dance action/steps in a defined ballet dance movement dictionary presents the possibility of training a CNN to identify specific ballet steps. The identification process should return a highly accurate output; therefore only movement with 90% accuracy should be acceptable. With the CNNs confidence level set to 0.9, a video of students performing a ballet enchaînement is checked and a transcript file of the identifiable defined ballet dance positions in the movement dictionary is created. The ability to transfer this transcript file to a usable Benesh Movement Notation (BMN) manuscript script could produce a useful educational tool and would decrease the amount of time required to create a BMN score for an entire ballet production.

Step identification and CNN output has the potential to be further developed with the intention of tracking the specific path of the body's limbs in the dancer's kinesphere.

ACKNOWLEDGEMENTS

I wish to thank:

My supervisor Dr Greg Doyle for his guidance, continued support and encouragement while undertaking this dissertation.

The Benesh International for the use of Benesh Notation Editor (BNE) lite.

The staff of Carlow IT Computer Services for their assistance in proving additional processing power.

The students of the National Performing Arts Studio (NPAS) in Dublin for allowing me to video them while they performed an enchaînement.

Colleagues and friends who posed for photographs.

I wish to acknowledge the assistance of Ms Martina Nunvarova in editing the 'Object_detection_video_transcript.py' and the coding of the 'translate_to_BMN.py' script.

Table of Contents

Table of Figures

RESEARCH QUESTION

It is intended to investigate if it is possible to use machine learning algorithms and image and video classification to identify a specific dance action/step in a defined ballet dance positions movement dictionary and to test this model on a recording of a performance by non-professional dancers. Subsequent to the achievement of this recognition, it is intended to attempt to display the dance action/step through a programming language, to Benesh Movement Notation (BMN).

1 INTRODUCTION

Machine learning has the ability to automate tasks for humans worldwide. In every day searches of the Internet, machine learning is at play through the algorithms used to return results geared towards previous search history. The current development of self-driving cars is another example of the use of machine learning. Machine learning Neural Networks are being trained to identify people, stop signs, traffic lights and other road signs. Machine learning is ubiquitous.

Benesh Movement Notation (BMN) is a dance notation language which is used for the representation and preservation of a ballet score. This language is also utilised in the syllabus books of the Royal Academy of Dance (RAD) to assist teachers in maintaining accuracy which may not be readily observed when using worded notation.

Using machine learning algorithms, visually capturing movement and transferring this movement in to a standardised notation score becomes a possibility. This could help reduce the time required to notate a ballet score.

The purpose of this study is to train a Convolutional Neural Network (CNN) to identify a number of predefined dance action/steps in a defined ballet dance positions in a movement dictionary. When the Neural Network is fully trained it is intended to test its accuracy on a recorded video of students dancing. Subsequently it is expected to convert and display the identified steps in BMN.

2 LITERATURE REVIEW

2.1 CHAPTER OVERVIEW

This chapter investigates BMN, its uses today and the available editing software. A discussion of Machine learning is also included. Neural Network algorithms, their architecture and the software libraries required to train them will also be discussed. An overview of current Data Protection legislation will be outlined.

2.2 BENESH MOVEMENT NOTATION (BMN)

While music is perceived through the ears, other factors, such as pitch, tempo and volume are also important (Forster, 2004, p.1). Music is represented through notation, using a treble and base clef, with five lines and four spaces for each clef. Movement is observed and interpreted through the eyes and the method of notation should visually represent the movement being carried out (McGuinness-Scott, 1983, p.139). Notation is an integral method of preserving the creator's chorographical intentions and conveying these clearly to the person intending to recreate them (Ryman et al., 1984, p.27). Ryman et al indicates that relying on memory alone for recall choreography may be prone to misinterpretation and errors in recollection (Ryman et al., 1984, p.27; McGuiness-Scott, 1983, p.139).

As movement itself is complex, accurately notating movement can be demanding. Many notation systems have been devised since the fourteenth century and some have failed to be effective in real circumstances, perhaps due to lack of sufficient explanatory detail (Ryman et al., 1984, p.27). One of the notation systems frequently used in ballet choreography was created by Joan and Rudolf Benesh in 1947 and is known as Benesh Movement Notation (BMN) (McGuiness-Scott, 1983, p.140). Originally BMN was designed to notate ballet, however it became evident that it was a method that could be used for notating all forms of movement (McGuinness-Scott, 1983, p.139). McGuiness-Scott (1983, p.139) considered that all people have the ability to read movement, providing it is presented in a perceptible manner. The notation that the Beneshes created was similar to that of an alphabet which could represent even the most complex of movements (McGuinness-Scott, 1983, p.139). However, in isolation it was initially not possible to decipher the meaning of the

alphabet. It was necessary to create rules for the formation of the grammar and the placement of the shapes on the notation stave (McGuinness-Scott, 1983, p.139).

In 1984, Ryman et al outlined the necessity for advancements in editing software. Although, notation systems have progressed, they lack computer graphical representation and automated editing software (Ryman et al., 1984, p.27). The University of Waterloo, Canada, has undertaken an interdisciplinary project, in dance and computing, with the aim of utilising a computer to record, edit, analyse and verify BMN with the aid of graphic techniques (Ryman et al., 1984, p.27).

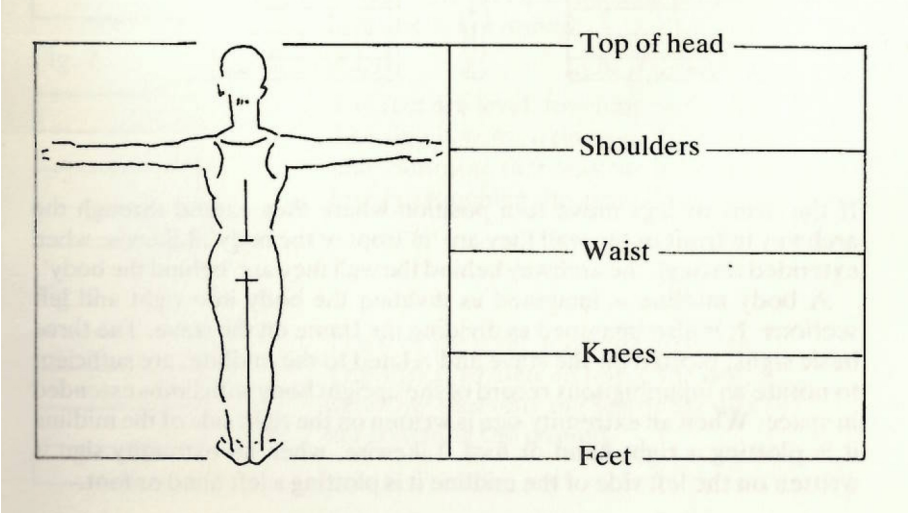

Figure 1. Demonstrates where the stave lines represent the body anatomical points (McGuiness-Scott, 1983, p.1)

BMN is written from right to left (Saad et al., 2012, p.1), the stave creating an ergonomic matrix for a human figure (McGuiness-Scott, 1983, p.1). The notation records movement from behind the figure (McGuiness-Scott, 1983, p.1; Ryman et al., 1984, p.28). Three symbols, indicating level, in front or behind, represent the relationship of the extremities as they relate to the rest of the body (McGuiness-Scott, 1983, p.2).

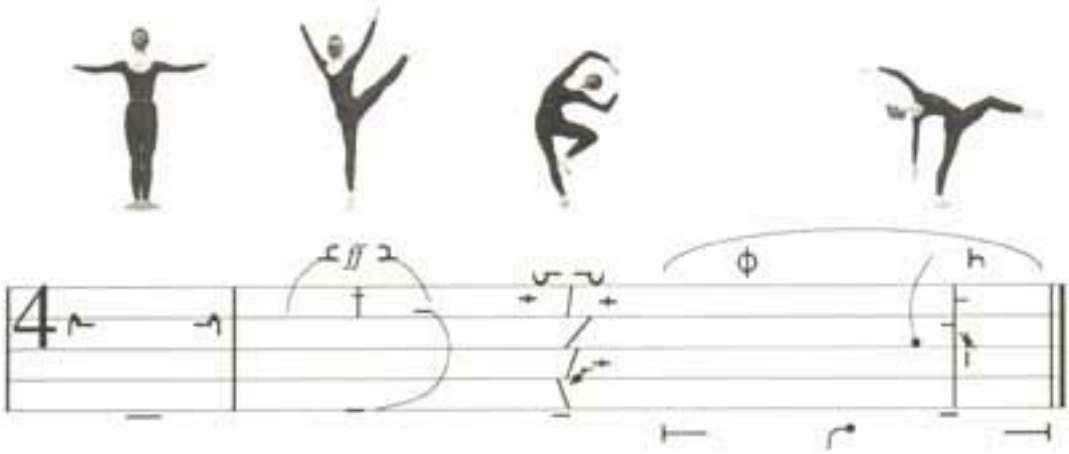

Figure 2. Example of BMN Score (Pinterest- BMN images, n.d.)

2.3 MACHINE LEARNING

Machine learning is a branch of Artificial Intelligence (AI) where investigation takes place to see how computers can improve their interpretation, understanding and, in turn, their reactions with constant iteration (Veloso, 2018). Data scientists Andreas Müller and Sarah Guido suggest that machine learning is about obtaining knowledge from data (Müller and Guido, 2016, p.1). Tom Mitchell of the machine learning department, Carnegie Mellon University, considers that any computer programme that can learn a task and can improve its performance through experience, is machine learning in action (Mitchell, 2006, p.1).

> A machine learns with respect to a particular task T, and performance metric P, and type of experience E, if the system reliably improves its performance P at task T, following experience E (Mitchell, 2006, p.1).

Machine learning is in operation when searching the Internet. However, it is not always obvious, as it is working in the background. The search engine is customised to a user's search history and interests and the search engine utilises this background data to produce results connected to the user's interests (Mitchell, 2006, p.1). Similarly, machine learning is used in filtering advertisements specific to the user, based on the user's historical data and preference (LeCun et al., 2015, p 436). Machine learning is also known as data mining, where big data is searched to discover patterns which are not immediately visible with the intention of making predictions (Géron, 2017, p.5).

According to Andreas Müller and Sarah Guido, Neural Networks can also be known as feed-forward Neural Networks or multilayer perceptrons (MPLs) (Müller and Guido, 2016, p.106). MPLs can be considered to be types of linear models that execute many layers of processing to yield a result (Müller and Guido, 2016, p.106). These linear models are executed over numerous steps, using an official procedure to reach a decision (Müller and Guido, 2016, p.106).

Aurélien Géron, head of YouTube's video classification from 2013 to 2016, suggests that machine learning is both the science and the art of providing a computer with the ability to learn from data through programming (Géron, 2017, p.3). Géron (2017, p.4) also considers that the definition of machine learning by Arthur Samuel's is a general definition. It is a "field of study that gives computers the ability to learn without being explicitly programmed". According to Géron (2017, p.4), Mitchell's 1997 definition is an engineering style definition.

Gary Bradski and Adrian Kaehler, contributors to openCV, suggest that the aim of machine learning is to change data into learning (Bradski and Kaehler, 2011, p.459). Once the machine has learned from the data provided, the expectation is that the machine can answer questions about the data. Using both rules and patterns that are present in the data, it is possible for machine learning to obtain information from the data (Bradski and Kaehler, 2011, p.459) and note similarities within the data (Bradski and Kaehler, 2011, p.459).

Machine learning structures can be categorised as supervised learning, unsupervised learning, semi supervised learning and reinforcement learning types. Categorisation occurs because of the amount of supervision received at the time of training (Géron, 2017, p.7). Supervised learning is used when filtering spam email. Unsupervised learning from previously gathered data can identify a group of similar visitors to a website (Géron, 2017, p.8). Semi supervised learning is utilised in Google Photos and requires some labelled data in order to label the rest of the data (Géron, 2017, p.13). Reinforced learning works on a reward based system; learning occurs through positive reinforcement, this is used frequently to aid robots when learning the mechanics of walking (Géron, 2017, p.14).

2.4 Supervised Learning

It is suggested that the most frequently used type of machine learning is supervised learning (LeCun et al., 2015, p.436). Zhi-Hua Zhou, head of the Department of Computer Science and Technology in Nanjing University, China, states that supervised learning can form a predictive model on labelled data which has been previously trained (Zhou, 2017, p.44). Training data consists of two elements, an example of an occurrence of something and a direct observation which is labelled (Zhou, 2017, p.44). There are the two types of supervised learning tasks which are carried out in machine learning regression and classification (Zhou, 2017, p.44). With regression supervised learning the labelled data has a numerical value, whereas with classification supervised learning the labelled data is categorical (Zhou, 2017, p.44). Zhou (2017, p.44) considers that the aim of regression supervised learning is to predict a quantity and the aim to classification supervised learning is to predict a categorical label. Ron Kohavi, engineer and general manager of analysis and experimentation team at Microsoft's Artificial Intelligence and Research Group, suggests that it is possible to estimate the accuracy of a supervised learning algorithm (Kohavi, 1995, p.1).

2.5 Unsupervised Learning

Unsupervised learning is a learning algorithm and is also known as clustering (Hahne et al., 2008, p.137; Müller and Guido, 2016, p.133). The algorithm extracts information from data that is input (Müller and Guido, 2016, p.133). Unlike in supervised learning, unsupervised learning does not require the creation of a training dataset (Hahne et al., 2008, p.137). However, as there is no training data set, unsupervised learning lacks the ability to predict the performance of the model through cross validation (Hahne et al., 2008, p.137). Hahne et al (2008, p.137) outline that clustering algorithms are assembled in terms of the best outcome, but there are no assurances that the best outcome will be achieved. A potential division of the data must be considered though (Hahne et al., 2008, p.137). This is not always possible even when dealing with an average sized sample of data (Hahne et al., 2008, p.137). In an attempt to gain the optimum output, different starting parameters are recommended (Hahne et al., 2008, p.137). Hahne et al (2008, p.137) discuss the requirements for the execution of unsupervised learning, and suggest: selection of

sample, selection of the attributes on which to perform the clustering, measurement of similarity within the sample and finally, algorithm selection (Hahne et al., 2008, p173).

2.6 Biological Neural Networks

ANNs and CNNs are based upon biological Neural Networks (Géron, 2017, p.257). The architecture of the brain was the system on which the infrastructure of Neural Networks was based (Géron, 2017, p.256). This was used to teach a computer how to learn (Géron, 2017, p.255). According to (Géron, 2017, p.258), in 1943, both Warren McCulloch and Walter Pitts put forward an uncomplicated model of a biological neuron (Géron, 2017, p.258). This model subsequently became known as an Artificial neuron, containing one or several inputs and one output (Géron, 2017, p.258). When a previously specified number of inputs are activated, the neuron activates its output (Géron, 2017, p.258). McCulloch and Pitts demonstrated the possibility of building a Neural Network with such a simple concept and design (Géron, 2017, p.258). This design would calculate any rational hypothesis (Géron, 2017, p.259).

2.7 History of the development of ANNs

According to Géron (2017, p.256), during 1943 and 1960, the development of ANNs progressed. In 1951, the first neurocomputer was developed by the mathematician Marvin Minsky and Dean Edmonds in Princeton University, New Jersey (Seising and Tabacchi, 2013, p.741). During 1957 and 1958, it was thought that the first computer for the classification of patterns was developed by both Frank Rosenblatt and Charles Wightman at Cornell University, New York (Seising and Tabacchi, 2013, p.741-2). Rosenblatt wrote an essay that described the first computer which contained the first Neural Network which was capable of learning and always returned the correct outcome. He called this machine Mark I Perceptron (Seising and Tabacchi, 2013, p.742). However, in 1969, Minsky and Seymour Papert, both mathematicians published their findings (MIT Media Lab, 2016; Seising and Tabacchi, 2013, p.742). It was not possible for ANNs of perceptron networks similar to that of Rosenblatt's to

deal with numerous individual instances (Seising and Tabacchi, 2013, p.742). As a result of these findings further investigation in this area was put on hold.

Approximately twenty years later, in 1980, there was a re-emergence of interest in ANNs (Géron, 2017, p.256). New architectures and improved training techniques were developed (Géron, 2017, p.258). In 1980, Kunihiko Fukushima, one of the pioneers in the Neural Network discipline, advocated the neocognitron as a hierarchical Neural Network model (Fukushima, 1980, p.193). The neocognitron was based upon a mechanism that recognises patterns and is modelled on how the human brain identifies symbols, letters of the alphabet and numbers (Fukushima, 1980, p.193). Fukushima highlights that the neocognitron Neural Network represents the model of a brain that is trying to learn patterns (Fukushima, 1980, p.193). The hierarchical Neural Network model utilises unsupervised learning (Fukushima, 1980, p.193). Fukushima states that the neocognitron model does:

> … not need any "teacher" during the process of self-organization [sic], and it is only needed to present a set of stimulus patterns repeatedly to the input layer of the network … the network acquires a structure similar to the hierarchy model of the visual nervous system proposed by Hubel and Wiesel (1962, 1965) (Fukushima, 1980, p.194).

In 1982, John J. Hopfield, biologist and professor of Physics at Princeton University, New Jersey, developed what is known as an Associative Neural Network and is also known as the "Hopfield Network" (Seising and Tabacchi, 2013, p.742). The Associative Neural Network algorithm is a learning-based algorithm that utilises Neural Network's long-term and short-term memory (Seising and Tabacchi, 2013, p.742). From 1986, psychologists, James L. McCelland and David E. Rumelhart, used ANNs practically to illustrate recognition of words when both spoken and seen (Seising and Tabacchi, 2013, p.742).

In the 1990s these developments further progressed with superior results (Géron, 2017, p.258). Support Vector machines were developed (Géron, 2017, p.258; Girshick et al., 2013, p.258). In 1998, LeNet-5 a CNN was introduced in a written paper by Yoshua Bengio, Yann LeCun, Leon Bottou and Patrick Haffer (LeCun et al., 1998, p.2278). The LeNet-5 was capable of classifying numbers that were hand written (LeCun et al., 1998, p.2278).

Both ANNs and Genetic Computing Programming are considered to be Soft Computing (Seising and Tabacchi, 2013, p.739). Soft Computing is deemed to deal with uncertain concepts, indefiniteness, disordered and muddled information (Seising and Tabacchi, 2013, p.739). In 2012, Krizhevsky et al produced considerably higher accuracies when working on image classification, which saw the re-emergence of CNNs (Girshick et al., 2013, p.1).

2.8 Neural Networks

An Artificial Neural Network (ANN) is made up of a great number of nodes that are interlinked, also known as neurons (O'Shea and Nash, 2015, p.1). The workload is distributed over the neurons, acquiring knowledge from the data that is input (O'Shea and Nash, 2015, p.1). This process returns an enhanced output (O'Shea and Nash, 2015, p.1). An ANN contains hidden layers and the learning process occurs in these hidden layers (O'Shea and Nash, 2015, p.1). The hidden layers make the decision about how a random change would affect, impair or enhance the ultimate output, based on the input from the previous layer (O'Shea and Nash, 2015, p.1). This process is known as the "process of learning" and the continuation of the process through numerous hidden layers is known as deep learning (O'Shea and Nash, 2015, p.1).

2.9 Convolutional Neural Networks (CNN)

CNNs are part of the Neural Networks grouping of algorithms (Müller and Guido, 2016, p.106; O'Shea and Nash, 2015, p.2). A CNN is a deep learning Neural Network based on the configuration of the biological brain and is a subset of machine learning (Sinha et al., 2017, p.1; Aloysius and Geetha, 2017, p.588). Deep learning is a grouping of methods that use a many layered structure to calculate presentation of data containing numerous layers of extraction (LeCun et al., 2015, p.436).

2.10 Architecture – ANN, CNNs

CNN's architecture is similar to that of conventional ANNs in that it is made up of neurons that self-develop through learning from data which has been input (O'Shea and Nash, 2015, p.2). CNNs are more focussed and capable of processing and

classifying images than ANNs (O'Shea and Nash, 2015, p.2). However, the architecture has reduced parameters because of its design; the architecture is encoded to process image-specific features (O'Shea and Nash, 2015, p.3). This focus is possible because of the architecture of the CNN, as it has been designed specifically to process pattern recognition within the input data (O'Shea and Nash, 2015, p.3).

A CNN contains three layers, namely: convolutional layers, pooling layers and fully connected layers as well as an input and output layer (O'Shea and Nash, 2015, p.4). Stacking of these layers builds a CNN's architecture (O'Shea and Nash, 2015, p.4). One of the advantageous characteristics of a CNN is the reduction in the quantity of factors that define the system and conditions in an ANN (Albawi et al., 2017, p.1).

The input layer of a CNN takes the image values of the pixels contained in the image (O'Shea and Nash, 2015, p.4). The convolutional layers are an integral part of how CNNs function and they ascertain the output of the neurons (O'Shea and Nash, 2015, p.5). Albawi et al (2017, p.5) highlight that the layer of paramount importance of a CNN is the convolutional layer. The majority of the calculation takes place in the convolutional layer (Sinha et al., 2017, p.1; Aloysius and Geetha, 2017, p.588). The architecture of a CNN is the location in which its functionality is found; the architecture permits the drawing out of characteristics at numerous levels (Sinha et al., 2017, p.1). When using CNNs in image classification, it is not important to find the specific location of the object within the image, as long as the object is identified (Albawi et al., 2017, p.1). In fact, problems that are to be processed by CNNs should not have aspects that depend on specific location (Albawi et al., 2017, p.1).

The neurons inside a CNN are ordered into 3 dimensions, namely: height, width and depth (O'Shea and Nash, 2015, p.4). This is the structural format of the input (O'Shea and Nash, 2015, p.4). Each layer contains neurons, but unlike in an ANN where each neuron in one layer connects with each neuron in the next layer, in a CNN each neuron connects to a small region in the proceeding layer (O'Shea and Nash, 2015, p.5).

A neuron is characterised by its weight, bias and activation function; it is obtained by working on an element at a time (O'Shea and Nash, 2015, p.6). The CNN layers

centre on kernels that learn through experience (O'Shea and Nash, 2015, p.6). When data arrives from the input layer at a convolutional, layer a 2D activation map is produced (O'Shea and Nash, 2015, p.6). The kernels operate when they see a specific feature in a particular place, these operations are known as activations (O'Shea and Nash, 2015, p.6). Each kernel will hold a corresponding activation map (O'Shea and Nash, 2015, p.6). These activation maps are located in the depth dimension (O'Shea and Nash, 2015, p.6). The activation map forms the entire output of the convolutional layer (O'Shea and Nash, 2015, p.7). The convolutional layer decreases the intricacy of the model as it uses the most effective method to reduce the resources used to manufacture its output (O'Shea and Nash, 2015, p.7). This reduction in the use of resources is achieved through the use of depth, stride and zero-padding hyper parameters (O'Shea and Nash, 2015, p.7; Aloysius and Geetha, 2017, p.588).

Depth refers to the depth created by the quantity of filters utilised per layer (Aloysius and Geetha, 2017, p.588). This quantity can be set manually by pre-setting the quantity of neurons (O'Shea and Nash, 2015, p.7). Reducing the depth can decrease the processing power required (O'Shea and Nash, 2015, p.7; Yang et al., 2017). However, it is important to note that reducing the depth can also reduce the accuracy and capacity of pattern recognition within the CNN (O'Shea and Nash, 2015, p.7).

Stride refers to the size of a step of a CNN (Albawi et al., 2017, p.3; O'Shea and Nash, 2015, p.7). It is possible for the user to set the size of the step manually (O'Shea and Nash, 2015, p.7; Albawi et al., 2017, p.3). Setting the stride sets the step size that the filter will move across the particular region (O'Shea and Nash, 2015, p.7). The higher the number of the stride, the greater the reduction in the amount of overlapping (O'Shea and Nash, 2015, p.8). This will also reduce parameters and return a smaller output (O'Shea and Nash, 2015, p.8).

2.10.1 PADDING

Setting the padding to zero captures all the data within an image and also reduces the size of the output (Albawi et al., 2017, p.3). Zero padding is used to manage the dimensional capacity of the output (Aloysius and Geetha, 2017, p.588; O'Shea and Nash, 2015, p.7). Zero padding can also be used to ensure all the information within

an image is checked (Albawi et al., 2017, p.3). If an object is present in the border region of the image, this will not be missed and can be checked by setting the padding to zero (Albawi et al., 2017, p.3). It is important to note that, if altering either stride or padding the output of the CNN layers will be affected (O'Shea and Nash, 2015, p.7).

A reduction in the number of connections and the ability to select the weights for the connections is the equivalent of sliding a window across the input image (Albawi et al., 2017). The neural input and the produced output makes it possible to identify characteristics without regard or consideration for their location within the image (Albawi et al., 2017).

The majority of the timeframe in the working of the interconnected layers of the system is used by the convolutional layers (Albawi et al., 2017, p.3). This is known as a network (Albawi et al., 2017, p.3). The execution of the task is dependent on the number of layers within the network (Albawi et al., 2017, p.3).

2.10.2 Pooling Layer

The main aim of pooling is to decrease the sample size of the data (Albawi et al., 2017, p.5). As a result, this would enable the reduction of the processing difficulties of the following layers within the CNN (Albawi et al., 2017, p.5; O'Shea and Nash, 2015, p.8). There are two types of pooling, namely: max pooling and general pooling (O'Shea and Nash, 2015, p.8). Albawi et al and O'Shea and Nash agree that max-pooling is the most frequently used (Albawi et al., 2017, p.5; O'Shea and Nash, 2015, p.5). This shortens the training time (O'Shea and Nash, 2015, p.8).

2.10.3 Fully-connected Layer

The fully-connected layer in a CNN is similar to the layout of the neurons in an ANN (O'Shea and Nash, 2015, p.8). Fully-connected layers link each neuron in a layer to each neuron in the next layer (O'Shea and Nash, 2015, p.8; Aloysius and Geetha, 2017, p.589). The fully-connected layer carries out tasks similarly to an ANN and attempts to produce results from the activations which will be used to classify data

within the image (O'Shea and Nash, 2015, p.8). The most widely used activation function is the Rectified Linear Unit (ReLu), the ReLu does not activate all neurons at the same time (Ide and Kurita, 2017, p.2648). A negative input will be converted to zero and any zero denoted neuron will not be activated (Ide and Kurita, 2017, p.2648). This means that at any one time only some neurons are activated making for efficiency and ease of computation (Ide and Kurita, 2017, p.2648). ReLu is now the standard activation function for Deep Neural Networks (Ide and Kurita, 2017, p.2648).

It is possible to manage the complexity of a Neural Network through the number of hidden layers (Müller and Guido, 2016, p.120). Generally when undertaking an image-centred pattern recognition exercise it will require classification which will utilise supervised learning (O'Shea and Nash, 2015, p.2).

2.11 Overfitting

Overfitting occurs when a Neural Network cannot learn (O'Shea and Nash, 2015, p.3). If overfitting occurs, there may be evidence of diminished accuracy in identifying general features in training, test and prediction data sets (O'Shea and Nash, 2015, p.3). CNN's reduced parameters and complexity requires less processing power to train and, as a result, reduces the likelihood of overfitting. When the occurrence of overfitting is diminished, the more likely accurate prediction can occur, thus improving the model (O'Shea and Nash, 2015, p.3). It is important to note that increasing the number of hidden layers, or the number of neurons, will not help reduce overfitting (O'Shea and Nash, 2015 p.3). It will only increase the amount of processing power and time required to train the Neural Network (O'Shea and Nash, 2015, p.3). It may not be possible to acquire the amount of processing power needed to train the Neural Network as unlimited computational power is not yet available (O'Shea and Nash, 2015, p.3).

2.12 Transfer Learning

It is not recommended to build a deep layered Neural Network from the beginning as it requires a significant amount of data (Géron, 2017, p.289). A dataset similar to ImageNet (ImageNet, 2016) or COCO (COCO - Common Objects in Context, n.d.)

datasets would be required and could take days to weeks to run (Géron, 2017, p.289). The best compromise is to find a Neural Network which has been previously trained to achieve a similar function to as the one currently required (Géron, 2017, p.289). This process reuses the lower layers of the CNN and retrains the top layer (Géron, 2017, p.289). Retraining the top layer is known as transfer learning (Géron, 2017, p.289). It is possible to find Neural Networks already trained at Model Zoos (Géron, 2017, p.294). Both TensorFLow and Caffe software libraries have their own Model Zoos, containing image classification and computer vision models (Géron, 2017, p.294).

2.13 Region-based CNNs (R-CNN)

Region-based Convolutional Neural Networks (R-CNN) are a modification of the CNN (Ren et al., 2016, p.1). When the R-CNN was first developed it was computationally expensive (Ren et al., 2016, p.1). The R-CNN used a method of selective searching within an image, rather than using the entire image to classify the object (Girshick et al., 2013, p.3). This made the R-CNN more proficient at object detection within an image (Girshick et al., 2013, p.2). This selective searching was put forward by Ross Girshick et al (Girshick et al., 2013). Image detection concentrates on the objects within an image; this concentration can be on multiple objects (Girshick et al., 2013, p.3). One method to detect objects within an image is to utilise a detector in the form of a sliding-window, similar to the method used in CNNs (Girshick et al., 2013, p.3). However, Girshick et al (2013, p.3) use "recognition using regions" to achieve object detection, a more suitable solution to object detection (Girshick et al., 2013, p.2). This method uses an algorithm to reduce the selective regions within the image to 2000 regions, and each region is classified (Girshick et al., 2013, p2). Girshick et al (2013, p.2) highlight three independent units that are combined in R-CNNs for object detection. Firstly, category-independent region proposals are generated, these regions are made available to the detector (Girshick et al., 2013, p.2). An R-CNN utilises selective search as it allows a controlled comparison with previously carried out observations (Girshick et al., 2013, p.3). Secondly, from each region a set length feature vector is extracted through a convolutional neural network (Girshick et al., 2013, p.2). Thirdly, a set of

standardised parts in the form of a Support Vector Machine (SVM) in sets of class linear SVM (Girshick et al., 2013, p.2).

Uijlings et al., (2013, p.3) discuss the selective searching algorithm's factors such as scale, diversification and computational time. As the size of objects within images vary and therefore, the outlining bordering is not always clear cut, this must be taken into account when searching for the object within an image in object detection (Uijlings et al., 2013, p.3). The search algorithm allows for this (Uijlings et al., 2013, p.3). Many factors, such as texture, lighting, parts omitted form picture and colouring can impact on the creation of an object, this diversification needs to be taken into account when searching an object (Uijlings et al., 2013, p.3). With this in mind the selective search algorithm does not utilise a single strategy, it has a multiple various searching strategy (Uijlings et al., 2013, p.3). This multiple search strategy uses a hierarchical algorithm, which clusters similar objects into groups (Uijlings et al., 2013, p.3). Girshick et al (2013, p.7) suggested that the architecture of an R-CNN can impact the accuracy of its detection. To reduce the occurrence of a bottle neck when running the search algorithm requires the ability to process quickly (Uijlings et al., 2013, p3).

2.14 Graphics Processing Unit (GPU)

Owens et al (2008, p.1) discuss the development of the graphics-processing unit (GPU), the sole purpose of which was 3D graphics. Even though it always had plenty of processing power, the challenge has been to expose this power to the programmers (Owens et al., 2008, p.2). Over time, the functionality of the GPU has evolved and is now powerful enough to take on taxing computational processing (Owens et al., 2008, p.1). The GPUs ability to take away complex tasks from the central processing unit (CPU) increases the ability of the final performance (Owens et al., 2008, p.2). The architecture of a GPU differs from a CPU; it is divided into space. The GPU utilises parallelism; one part of the processor works on a particular stage and when it is finished, it passes it onto the next part (Owens et al., 2008, p.3).

2.15 FAST R-CNN

Shaoqing Ren co-founder of Momenta in China, developed autonomous driving software with the aim of progressing technology so that self-driving cars become a reality (Ren, 2016). Ren et al (2016, p.1) highlight that the Fast R-CNN algorithm has reduced the time frame when running object detection; this can be achieved through the extra processing power that is created with the use of GPUs.

Ros Girshick, is a research scientist at Facebook AI whose area of interest is computer vision and machine learning (Girshick, n.d.). Girshick (2015, p.2) highlights that a Fast R-CNN was designed to address the design flaws of R-CNNs. The Fast R-CNN has the ability to train a deep detection network nine times faster than an R-CNN (Girshick, 2015, p.1). This is achieved by the use of a new training algorithm, an algorithm that has a greater detection standard than is used with R-CNN (Girshick, 2015, p.2). The Fast R-CNN uses a multi-task loss in a single-stage training; all of the network layers can be updated by training and less disk space is required as none is needed for feature caching (Girshick, 2015, p.2). A Fast R-CNN can take an image that is input as a whole and an object proposal which will identify the areas in the image where the object is located, regardless of the category (Girshick, 2015, p.2). Girshick (2015, p.8) presents the Fast R-CNN as a quicker reconditioning of the previous R-CNN. Girshick (2015, p.8) considers Fast R-CNN as a "clean and fast update to R-CNN". Ren et al (2016, p.1) state that Fast R-CNN's have decreased the amount of time taken to process object detection networks.

2.16 FASTER R-CNN

Faster R-CNN is an advance on its predecessor, the Fast R-CNN. Faster R-CNN's aim is to focus on object detection in real time (Ren et al., 2016, p.1). Ren et al (2016, p.3) highlight that there are two sets of standard parts which are known as modules, used in the object detection system. Firstly, a deep fully convolutional network which presents the regions to the second module (Ren et al., 2016, p.3). Secondly, a Fast R-CNN detector, which utilises the proposed regions form the convolutional network (Ren et al., 2016, p.3). The Faster R-CNN's structure is a united fusion of neural networks with an addition of a Region Proposal Network (RPN) module (Ren et al., 2016, p.3). A RPN takes an input image of any size and outputs a regional score to

each object detected by assigning the object to a particular class (Ren et al., 2016, p.3). The RPN then indicates where to check to the Faster R-CNN (Ren et al., 2016, p.3). In order for the two models to work coherently and efficiently together, both the RPN and the Faster R-CNN need to share the training of their layers rather than work independently (Ren et al., 2016, p.5-6). Ren et al (2016, p.12) detail how the Faster R-CNN is a cohesive deep neural network capable of object detection at almost real-time frame rates. The use of the RPN increases the regional detection within an image and therefore increases the accuracy of detection.

2.17 INCEPTION

Christian Szegedy et al (2016, p.2818) outlined that Neural Network Inception was designed and built to function fully with restrictions on the amount of memory that is available to process. Inception utilises significantly less parameters than both AlexNet and VGGNet (Szegedy et al., 2016, p.2818). Inception is the ideal choice for the processing of big data situations (Szegedy et al., 2016, p.2818).

2.18 CNN – INCEPTION-V3 CNN

In 2014, Szegedy et al (2015, p.2818) stated the architecture of the CNN had improved due to the increase in both depth and width of the network. The resulting architectural improvements led to better results from digital images or videos (Szegedy et al., 2015, p.2818). The Inception model's architecture was robust in performing well with only a small amount of memory and computational availability (Szegedy et al., 2015, p.2818). Because Inception can operate with limited computational capacity, it is useful in visual computer processing activity where a large amount of processing capacity is required (Szegedy et al., 2015, p.2818).

2.19 IMAGENET DATASETS

ImageNet is an image database on which Inception-v3 CNN was originally trained (ImageNet, 2016). In the dataset there are four synsets of ballet; ballet master, ballet dancer, ballet skirt/tutu and mistress. Ballet master is described as "A man who directs and teaches and rehearses dancers for a ballet company"; master contained three hundred and seventy-four images. (ImageNet Ballet Master, 2010) Ballet

dancer is described as “A trained dancer who is a member of a ballet company” and contained one thousand four hundred and sixty eight images (ImageNet Ballet Dancer, 2010). Ballet skirts/tutus is described as “very short skirt worn by ballerinas” and contained one thousand four hundred images (ImageNet Ballet skirt, tutu, 2010). Ballet mistress is described as “a woman who directs and teaches and rehearses dancers for a ballet company” and contained one hundred and ninety four images (ImageNet Ballet mistress, 2010).

2.20 Faster R-CNN – Inception-v2 Faster R-CNN

The Faster R-CNN is considered to be the most advanced algorithm of object detection (Chen et al., 2018, p.4). In 2017, Sun et al (2017, p.4) discussed the Faster R-CNN’s advanced ability in object detection. The Faster R-CNN is a two-step model, a combination of Regional proposal network (RPN) which uses the selective search combined with the architecture of the Convolutional Neural Network (Sun et al., 2017, p.4; Shen et al., 2018, p.1). Due to the two step sharing of the convolutional layers, the architectural capability of the Faster R-CNN leads a lessening in the required processing power (Chen et al., 2018, p.4; Ren et al., 2016, p.1).

2.21 COCO datasets

The Faster R-CNN Inception-v2 Common Objects in Context (COCO) was originally trained on the COCO image dataset (COCO - Common Objects in Context, n.d.). The images were sized at 244 pixels by 244 pixels in order to train the images. The COCO dataset labelled the images with great accuracy, the item requiring identification was only outlined and labelled, unlike using labelImg where a square is drawn around the identifiable (classified) object, see Fig 3. It is evident that the labelling used in the COCO dataset is much more accurate than the box outline used in ImageNet.

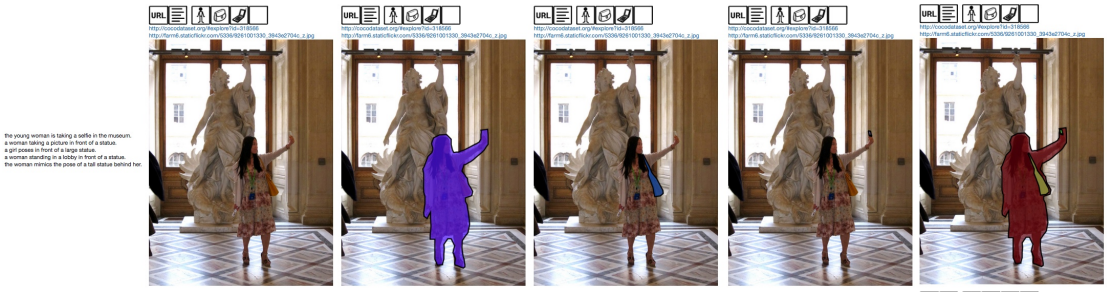

Figure 3. Label process used by COCO dataset

2.22 Deep Learning Libraries

2.22.1 OpenCV

In 2000, OpenCV (Open Computer Vision) was released by Gary Rost Bradski, principal engineer and manager of Machine Learning group for Intel Research, Adrian Kaehler, a senior software engineer and Vadim Pisarevsky, a software engineer (Goldsborough, 2016, p.2; Bradski et al., 2005, p.119). OpenCV is an image processing library and is designed to resolve learning tasks in computer vision and image recognition (Goldsborough, 2016, p.2). Pre-defined classes are created containing objects the user wants to identify. Ordinarily, when detecting objects in a static image or a video frame, it is ideal to extract attributes such as edges, colour regions and textures as well as using problem solving methods to identify attributes that are key to the object being investigated (Bradski et al., 2005, 119). Due to the irregularity of human faces; some people wear glasses; some have beards, lighting will also change the appearance of a face (Bradski et al., 2005, p.119). As a result, when trying to detect human faces, it is difficult to find attributes and problem solving methods that will encompass the vast number of variations present within the object class (Bradski et al., 2005, p.122). A classifier, a statistical model, can be trained using both images that contain the object of interest and images that do not contain the object of interest (Bradski et al., 2005, p.122). The classifier is then used to reveal the object from it's predefined class (Bradski et al., 2005, p.122). OpenCV uses the classifier and statistical model for object detection (Bradski et al., 2005, p.122). A fixed image size is used to train the classifier; the classifier has the ability to scale the image size (Bradski et al., 2005, p.122). The image is combed using the fixed size to scan the image and checked to see if an object or a face is present in that particular location (Bradski et al., 2005, p.122).

2.22.2 TensorFlow

Google released TensorFlow, a machine learning open source library in November 2015 (Géron, 2017, p.231). TensorFlow is considered to be one of a few functional frameworks for training deep learning models (Goldsborough, 2016, p.1). It provides the building blocks when building, training and validating a Neural Network (Seo et al., 2018, p.1). TensorFlow operates in a heterogeneous environment, incorporating the ability to function fully using both hardware and software from different service providers (Abadi et al., 2016, p.265). TensorFlow represents machine learning algorithms graphically in the form of computational graphs (Goldsborough, 2016, p.3). Also known as dataflow graphs, computational graphs are a type of directed graphs (Goldsborough, 2016, p.3). Directed graphs produce a graphical representation of a machine learning algorithm where objects, called vertices or nodes describe operations (Goldsborough, 2016, p.3). Operations in TensorFlow are an expression of data flowing through the graph (Goldsborough, 2016, p.3) The objects are connected together through links, referred to as edges, that direct data from one object to another (Goldsborough, 2016, p.3). Edges in TensorFlow are known as tensors (Goldsborough, 2016, p.3).

2.23 Software Tools

Software engineer Darrenl Tzutalin, who has experience with computer vision and machine learning, created labelImg (tzutalin (darrenl), n.d.). LabelImg is a tool that can create an object bound box on an image. This image graphical tool is similar to the format that ImageNet used when classifying images. For its graphical interface LabelImg uses Qt and is written in Python (darrenl, 2018). LabelImg outputs/produces a xml file, this xml file details the file path of the image, the image name, the height, the width, the number of the RIB channels, the label allocated to the image and the location of the bounding box of the label.

2.24 Action Detection and Recognition of Human Movement in Video

Gorelick et al analysed actions as space-time shapes and considered that the resulting representation provides illustrative data about the analysed actions (Gorelick et al., 2007, p.2247). The ability to identify human action from video is an integral part of movement analysis. The strategy applied by Gorelick et al (2007, p.2247).involves deduction of a human action that contains both spatial and dynamic information The spatial information reveals the direction and the location in which the figure and their extremities are within space. The dynamics refer to the person's body actions as a whole, and also to their extremities in relation to their body as a whole when walking or running. Gorelick et al are using what is known as 'space-time shape' (Gorelick et al., 2007, p.2247). If there is adequate detail in the 2D image, it is possible to extract information indicating the action that the body is carrying out (Gorelick et al., 2007, p.2247).

Figure 4. Space-time shapes of "jumping-jack", "walk" and "run" actions.

In the case of actions that are similar, but not exactly the same, a different space-time intensity volume will be detected and produced (Shechtman and Irani, 2005, p.1). This difference occurs because of difference in clothing and in backdrop (Shechtman and Irani, 2005, p.1). As people themselves are different shapes, this also applies to their clothing (Shechtman and Irani, 2005, p.1). Shechtman and Irani (2005, p.1) investigated a method to measure this without focusing on either the foreground segmentation, background segmentation or on the activities carried out by the individual. When the research of Gorelick et al first appeared at the Tenth IEEE International Conference on Computer Vision in 2005, it was indicated that the Poisson equation was used to identify space-time features (Gorelick et al., 2005, p.2247). However, due to the progressive nature of Machine Learning, Neural Networks have provided an alternative method of identifying actions in space-time.

2.25 General Data Protection Regulation (GDPR)

In April 2016 the European Union (EU) General Data Protection Regulation (GDPR) was endorsed. GDPRs aim to coordinate twenty-eight different data protection laws in place across EU. The concepts and principles of the GDPR are like the Data Protection Acts of 1988 and 2003. However, the new GDPR requires accountability when processing personal data, particularly the processing of children's data. A company must have consent from a parent or guardian, if the young person is under the age of sixteen. When dealing with under age customers, the participants must understand to what they are giving their consent. It is the responsibility of the company to use language which children understand (Data Protection Commissioner, 2018). The new GDPR came into effect on the 25 May 2018 (Data Protection Commissioner, 2018). One of the aims of the change in legislation was to reduce the number of compliances required by different authorities; this was to help business within the EU adhere to a set of coherent rules that were in place across the EU.

The new law stipulates that consent obtained from children who are under the age of sixteen, is only valid if accompanied by permission from parent 31/07/2018 10:46 In some member states the consent age of sixteen may be reduced to thirteen years of age (EU General Data Protection Regulation, 2016). The new GDPR outlines that

valid consent must be given freely and the function of the data should be explicitly stated. Any data that has been gathered must be documented and it must be clearly outlined that the participant has given permission (EU General Data Protection Regulation, 2016). Participants who posed for photographs were above the age required for GDPR compliance with regard to seeking parental or guardianship permission. Written and oral GDPR compliance requirements were fulfilled regarding this group. See Appendix A and B.

3 METHODOLOGY

3.1 CHAPTER OVERVIEW

In this chapter the gathering of the data is discussed as well as the processing of the Inception-v3 CNN and the Inception-v2 Faster R-CNN. Also included is the labelling process used, the testing of the CNNs and the process of creating a transcript to be converted to BMN.

3.2 GATHERING DATA TO FORM DATASET

Images were downloaded from Google images; videos were downloaded using YouTube Multi Downloader Online (Downloader, n.d.). Videos were downloaded from the Royal Opera House's YouTube page (Royal Opera House, n.d.). Also, videos from the Royal Academy of Dance syllabus were used. Volunteer participants also posed in ballet positions. The three different methods of image collection, Google images, YouTube video photographs and volunteer photographs, captured various images from different perspectives and levels of ballet position and technique. The videos were processed using a Python Script. This script extracted the images per frame. Initially, the Python script was set to extract every frame of the video. However, an unnecessarily large number of images were produced. The script was then amended to produce an image of every twenty-fourth frame. Again, an unnecessarily large number of images were produced. The intention of the frame extraction was to capture the dancers in different ballet positions as they passed through different movements of an enchanîment. This frame extraction was to assist in the gathering of data for the training of the Neural Network. Some of these images were used for classification where dancers were transitioning from one step to another. Because of this transitioning, the dance position in the image was not always exact, and therefore the image was not always clear-cut. This variation in the images is an advantage in Neural Network training as being exposed to variations allows a more rounded interpretation of the image. Foot placement of the same position varies from person to person. This exposure to the various placements of the foot gives the Neural Network an opportunity to learn from different viewpoints.

3.3 BUILD OR PRE-BUILT CNN

Having researched and investigated the area of Machine Learning, including both supervised and unsupervised learning algorithms, ANNs and CNNs, it seemed appropriate to use a CNN for the purpose of this research. While ANNs are composed of neurons that learn through iteration from input data, CNNs are superior when dealing with image classification. For the purposes of this research, two options were available regarding the choice of CNN – to build de novo or to retrain the Neural Network. In 2017, Bowen Baker et al., (2016 , p.1) suggest that the process of designing a CNN is not only labour intensive but also requires expertise. Because of time constraints and a lack of familiarity and expertise in working with these algorithms and also the possibility that the building of the CNN might not be successful, it was decided to retrain an existing CNN. Transfer learning was used when training only the last and final layer of the CNN. By only retraining, the last layer of a CNN forms an accurate classification in a shorter amount of time.

Various individuals and companies such as Google have developed and built CNNs. Having researched a number of CNN algorithms, GoogLeNet's Inception-v3 was chosen for the suitability of its architecture and its efficiency and ability to cope with the limitations of the processing power available. The choice of Inception-v3 was made because of its ability to fulfil the requirements of this research, namely, pattern recognition of movement through machine learning. Inception models use less processing capacity and power to that of VGGNet (Szegedy et al., 2015, p.2818).

3.4 CNN INCEPTION-V3

3.4.1 PARTIALLY RETRAIN CNN – TRANSFER LEARNING

As the last layer of a CNN Inception-v3's top layer can be retrained to the specific ballet dance positions in the movement dictionary of dance steps; initially it was decided to retrain the top layer only of the CNN Inception-v3. The Inception-v3 dataset was trained using some images that contain ballet, but not on specific movements. The pre-trained Inception-v3 model is loaded. All the images are analysed returning an image vector which will be used to train the top layer of the CNN. The lower layers of the CNN are already trained in how to distinguish between

objects. This can be used without alteration in recognition tasks. The transfer learning process does not require a large amount of processing power as it is only training the top layer of the CNN and it utilises the previously taught differentiation techniques.

3.4.2 Installing Libraries CNN

Having researched the available machine learning algorithm libraries, the decision was made to use TensorFlow as it best fulfilled the requirements of this research, image and movement classification. Pip was installed first. Pip can be used to install TensorFlow's hub. TensorFlow was installed using pip on the command line. TensorFlow's hub retrains the top layer using four thousand iterations. This can be manually altered

3.4.3 Classifying Movement CNN

In order to compile a dataset it was necessary to create a framework of movements, a method of automating movements that connects with BMN. Use was made of the concept of the likelihood of the next step, as there are only so many steps that can come from a static position. In this fashion, the computer can learn the predictability of the proceeding or oncoming movement. To facilitate this, the movements were broken into potential classifications the defined ballet dance positions in the movement dictionary of movements. Rather than creating a dictionary of all the potential positions and steps that could possible occur, it was decided to limit the number of steps classifications that the computer would be trained on. The defined ballet dance positions movement dictionary was created. This number could be increased once the algorithm was able to sufficiently identify the correct step/dance action at a later time.

3.4.4 The process for Inception-v3 CNN

In order to use Inception-v3 it was necessary to download TensorFlow. Inception-v3 CNN was downloaded. It was not necessary to download the Inception-v3 model as the commands issued used when using TensorFlow's hub accessed an online version

(hub: A library for transfer learning by reusing parts of TensorFlow models, 2018). If the use of a different Inception CNN model is required, this can be achieved by change of URL. Instead of retraining the entire Inception-v3 CNN, only the top layer was retrained. The size of the image does not impact on the training of the CNN model; if the image is reduced too much it will not return as accurate results that are possible. TensorFlow's hub automatically creates two datasets; a training dataset and a test data set.

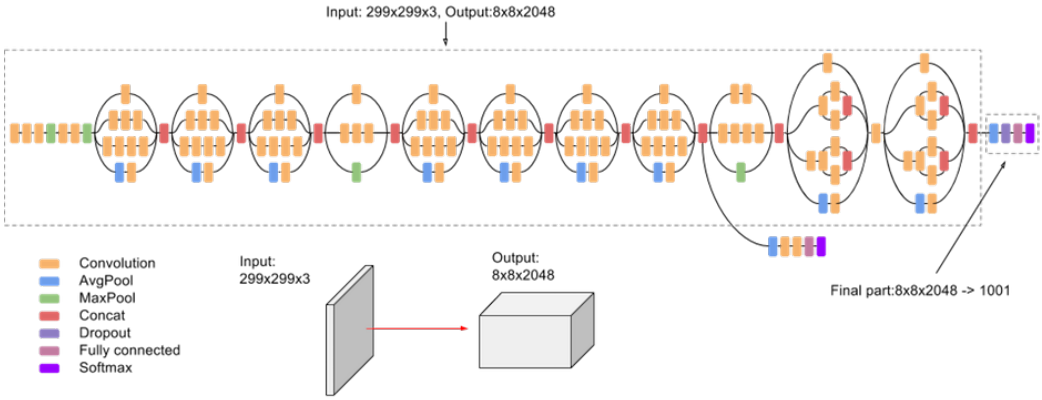

Figure 5. Inception-v3 CNN architecture

The images were placed in eight different subfolders, labelled by the name of the dance step in the defined ballet dance positions in the movement dictionary. The folders were called '1st', '1st_arms', '2nd', '2nd_arms', '5th', '5th arms', bras_bas' and 'plie'. There was no sub folder for training and test data. In the retrain.py Python script the data is split into training and test photos at random.

The last layer of the model was retrained on the newly created datasets. The model was unable to accurately classify many of the correct positions. It was found that one of the possible reasons for the inaccuracies was misclassification of the data. Some of the images were misplaced, having been placed in the incorrect folders. The data was rechecked and the model was retrained. This returned an improved version of image classification of the defined ballet dance positions in the defined ballet dance positions in the movement dictionary. However, further refinements are required as inaccuracy is still evident.

3.4.5 DATASETS CNN

Initially, a ballet dance positions movement dictionary was created of a small number of basic ballet positions, namely; first, second and fifth positions of the feet, bras bas, first, second and fifth position of the arms as well as the action of a demi plié. Choosing a small number of positions and dance actions for the ballet dance positions in the movement dictionary at the beginning of the research, has the advantage of using a lesser amount of processing power than when using a larger dictionary of ballet steps, thus reducing the time spent on training and testing the tailored Inception-v3 CNN.

Originally, the intention was to gather data from Google images and YouTube. This was carried out for dataset one and dataset two. Dataset one contained gathered random images from Google images and sections of videos, which were converted, to jpg frames. The movements were classified by creating a folder for each step within the defined ballet dance positions in the movement dictionary.

Dataset two contained images that were taken from Google images and sections of videos which were converted to jpg frames. These images were cropped to identify only the specific body part, feet or arms, required for classification. Some of the cropped images were pixelated and blurry. This inability to get clear, accurate pictures may have meant that the Inception-v3 CNN could not process the images and may have caused an error to be returned. Subsequently, the pixelated images were removed, reducing the number of images present within the defined ballet dance positions in the movement dictionary folder and thus permitting the Inception-v3 CNN to retrain. However, when the Inception-v3 CNN was tested, the return was inaccurate. TensorFlow accepts images of various sizes. Images used in the retraining process can be various sizes. The retrain.py script automatically sets the size of an image's height at 244 pixels and its width at 244 pixels. This size can be amended in the retrain script. However, the size of the image does impact on the accuracy of the final output. Smaller images produce less accuracy, but use less processing power.

Dataset three contained images of volunteers who agreed to being photographed when posing in the above mentioned defined ballet dance positions in the movement dictionary of ballet feet and arm positions. All volunteers completed a consent form indicating their willingness to participate in the research, with the proviso that the dataset would not be published on the Internet, as some had reservations about the use of their facial image on this medium. When the Inception-v3 CNN was retrained and was tested, the return was slightly more accurate than dataset two, but was not a satisfactory classification.

Dataset four was a combination of datasets one, two and three. Again, there was a slight improvement with the classification results, but only one image was classified with an over 80% certainty, this still was not a satisfactory classification.

Dataset five was comprised of dataset four, but some of the same pictures were additionally represented on a superimposed background. For example, some of the participants were placed on different background. The results returned represented a reduction in accuracy when compared with dataset four, particularly the classification of the feet in a first position. Again this was not a satisfactory result.

Dataset six was comprised of dataset four and new images. The images on the superimposed background were removed from dataset five. New images found online and were added to balance the number of images taken and the random images. The image folders were checked for any misclassified images. The accuracy of dataset six represented a small improvement when compared to the accuracy results of dataset five.

Dataset seven was comprised of dataset six and new images. Photographs were taken of two new volunteers who gave their permission to pose for photographs in the required ballet feet and arm positions. These volunteers completed a consent form. The photographs were taken with two different backgrounds; one was in a gym setting the other in a corridor with a slightly patterned floor. These different backgrounds may not have helped with the image classification. The results returned from this classification generally indicated a lower level of accuracy than previous datasets, but did identify the feet in second position.

Dataset eight was comprised of dataset seven. Some of the images were cropped to focus more on the specific position. For example, the image was cut to see the waist to the ankles to display the demi plié. The results returned from this classification was similar to results of dataset seven.

3.5 FASTER R-CNN FASTER-R-CNN-INCEPTION-V2 TRAIN FASTER R-CNN

3.5.1 INSTALLING LIBRARIES R-CNN

In order to train the Faster R-CNN Inception-v2 while using a Mac OS laptop, it was necessary to download TensorFlow, Anaconda, Python 3.6 and OpenCv as these were not preloaded.

Using a gaming HP 'ENVY' laptop with a graphics card, Windows 10 64-bit Operating System, 1.6GHz Intel Core i5-4200U, 4GB RAM with NVIDA GeForce GT 740M with graphics memory of 2GB. TensorFlow 1.8, CUDA 9.2.88, CUDNN v7.1 for CUDA 9.2, TensorFlow GPU Anaconda and Python 3.6 were downloaded. The downloaded TensorFlow GPU version required matching the CUDA and CUDNN library's compatible.

However, at only step 1277 the loss had increased to 7121406.0000 instead of reducing.

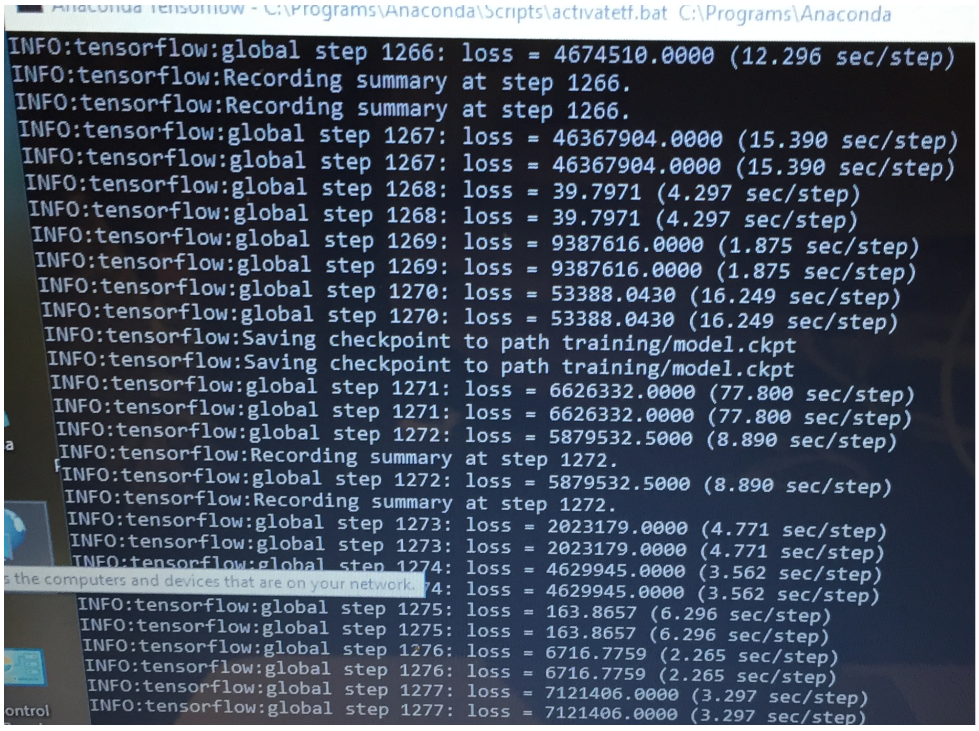

Figure 6. Faster R-CNN training on HP 'ENVY' Gaming laptop

Using a DELL desktop computer with Windows 7, TensorFlow GPU was downloaded. This computer had a graphics card and as a result it was possible to use TensorFlow GPU. TensorFlow was installed via Anaconda. The Windows 7 desktop, 64-bit Operating System, 3.4 GHz Intel Core i7-2600, 8 GB RAM CPU contained an AMD Radeon HD 6900 Series graphics card. Unfortunately, the Radeon graphics card was not compatible with CUDA and as a result, TensorFlow GPU was not operational.

The Mac book Pro, OS High Sierra, 2.9 GHz Intel Core i7, 16 GB RAM and an Intel HD Graphics 400 1536MB device was used to train both the CNN and the Faster R-CNN.

3.5.2 The process of labelling the body parts for classification R-CNN dataset nine and ten

Before labelling the images, it was necessary to decide how many movements would be in the ballet dance positions in the movement dictionary of movements that the algorithm was expected to classify correctly.

As it was the second time retraining the Faster R-CNN-Inception- algorithm, it was best to refine the previously tried methods and start the process with a smaller number of static position movements. The method used to label the images was similar to the concept of BMN. For dataset ten the ballet dance positions in the movement dictionary consisted of twelve classifications. The body was broken into regions, the feet, the knees and the arms.

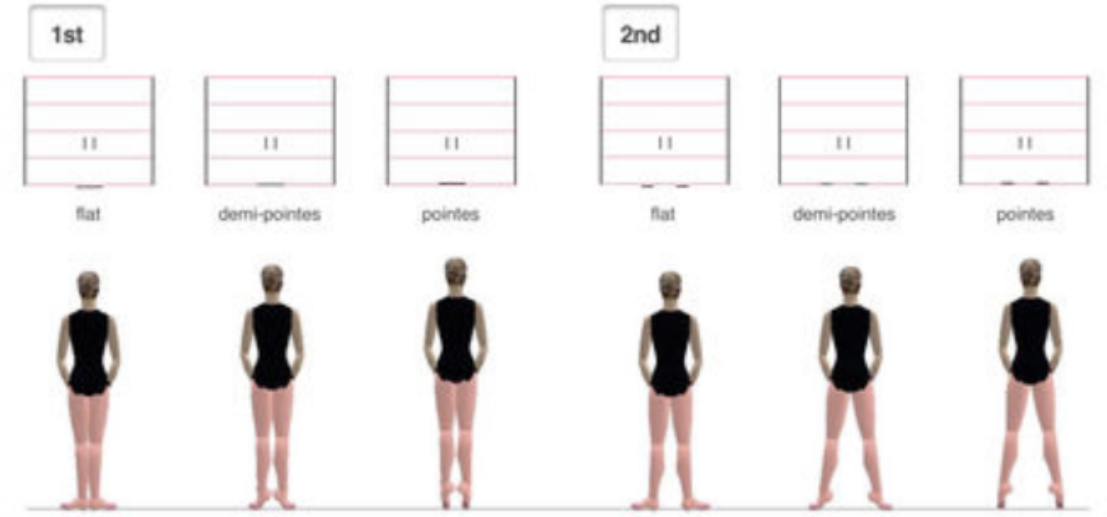

Figure 7. (Ryman et al., 1984)

THE FEET

In ballet, feet level position can occur at three different levels – feet entirely on the floor, demi pointe with ball of foot and toes in contact with the floor and en pointe where only the tips of the toes are on the floor. Bearing in mind that the majority of teachers wanting to use this tool would be concentrating on capturing the choreography of the younger students, en pointe is not of paramount importance at this stage. Also, the students dancing in the test video are not dancing en pointe. In the dataset, the levels were labelled 'flat', 'demi_point' and 'point'.

In ballet there are five basic positions of the feet. In the case of this research, three of these positions are being used: first where the feet and legs and feet are rotated away from the mid-line of the body and the heels touching, second where legs and toes are rotated away from the mid-line of the body but there is a gap between the heels which are just under the hips, fifth position both legs and feet are rotated away from the mid-line of the body, with either the right or left leg in front, the placement of the front foot is the heel of the front foot to the head of the metatarsal bone. Fifth position will be broken into two categories, fifth with the right leg in front and fifth with the left leg in front.

THE KNEES

For the purposes of this research the knees are either bent or straight. In the dataset, the knees were labelled 'demi_plie' and 'straight_legs'.

THE ARMS

In ballet, there are six basic positions of the arms. For the purposes of this research, three of these positions are used: bras bas, first, and second positions of the arms. Bras bas arms are rounded with the little fingers in line with the top of the thighs. In first position, the arms are in the same rounded position, but outstretched in front and in line with the navel. In second position, the arms are in the same rounded position but opened to the side of the body, palms facing forward, with a gentle sloping downwards. In the dataset, the positions of the arms were labelled 'bras_bas', '1st_arms' and '2nd_arms'.

When labelling the images using labelImg (tzutalin, 2018) it is was necessary to outline the particular areas of the body to allow the algorithm to correctly identify the correct position. For the levels, the entire foot to the ankle was labelled with a box for each foot. For the position of the feet, one box was created from above the ankles to the tips of the toes. For either bent or straight legs, above the hips to the ankle was outlined with a box. For the positions of the arms, the top of the head to the tips of the fingers was outlined and labelled with a box. It was important to accurately label the positions and not to include too many other body parts, otherwise when trying to classify, the algorithm would look for too much information and misclassify, as it expected the hips to be in a specific placement. It is important to exclude non-relevant items in the box.

3.5.3 THE PROCESS FOR INCEPTION-V2

Each image in the dataset was labelled using LabelImg software. The dataset was then divided into two types; training and test data. The majority of the data is placed in the training dataset. It is split approximately 70% in the training dataset folder with 30% in the test dataset folder. This split can vary between 80% to 20% or 90% to 10%. Returning the most accurate results depends on trial and error. It is important to have a number of images each of the ten positions in the defined ballet dance positions movement dictionary within the testing folder.

3.5.4 Acceptable image format CNN and R-CNN

It is important to note that images in the png file format are not processed when training the CNN. When using Inception-v3, no errors were returned but the images were not processed. When using Faster R-CNN-Inception-v2, any files that contained a png file format returned an error. The images were converted to jpeg file format to prevent any images being lost.

3.5.5 Faster R-CNN Inception-v2

First Time Running

All the images were catalogued using the labelImg, labelling software. All images are gathered and labelled; dataset nine was used to retrain the Faster R-CNN Inception-v2 COCO (TensorFlow Models: detection_model_zoo.md, 2018). Ten labels were created: straight, en pointe, demi-plié, bras bas, 5th_r, 5th_l, 5th_arms, 2nd, 2nd_arms, 1st and 1st_arms. Some of the data was misclassified. Errors were returned as the French spelling of the names forced the Faster R-CNN to crash before it would start to run. As a result some of the images were misclassified in dataset nine.

The R-CNN ran on dataset nine for over forty hours. The minimum time it took to process an iteration was approximately 2.402 sec/step and the maximum amount of time was 13.695 sec/step. The average time per sec/step was approximately 5.0 sec/step. Running the Faster R-CNN – Inception-v2 for forty hours it had not dropped to the required consistent below 0.05. It ran 17639 steps the loss was 0.0155 (9.960 sec/step), however the step before the loss was 0.2389 (11.488 sec/step), it had still not run long enough in order to create an even loss for the model to accurately identify the specific positions. Even though the accuracy had not dropped consistently below 0.02, the model was able to correctly identify '5th_r' – the ballet position 5th with the right leg in front. It also identified the feet as first position. Re-checking the input data, some of the images were misclassified. As they were misclassified it was not a valid reflection on the output results.

Second Time Running

The dataset was much tidier on the second occasion it was again divided into twelve ballet positions in the defined ballet dance positions in the movement dictionary. The

Faster R-CNN Inception-v2 ran for over one hundred and eleven hours. The minimum time it took to process a single iteration was approximately 3.185 sec/step and the maximum amount of time was 26.695 sec/step. The average time per sec/step was approximately 5.751 sec/step. Running the Faster R-CNN – Inception-v2 for one hundred and eleven hours the losses not dropped to the stable consistent below 0.05. It ran 77436 steps the loss was 0.0852 (5.276 sec/step), however the step before the loss was 0.27033 (5.481 sec/step), it had still not run long enough in order to create an even stable loss for the model to be fully training.

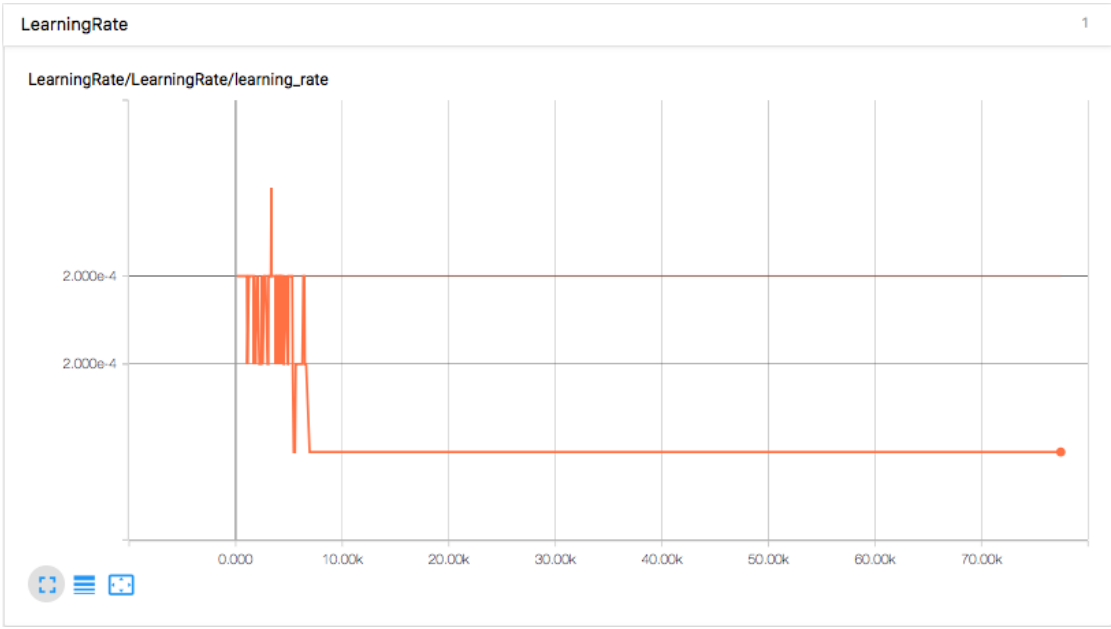

Figure 8. Learning Rate of Faster R-CNN - Second Time

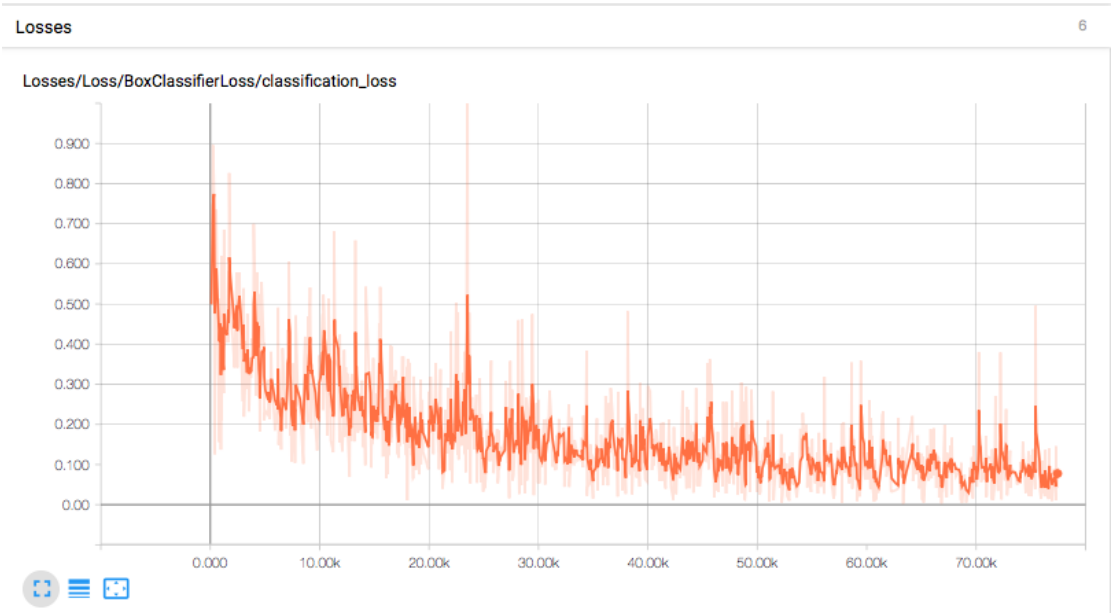

Figure 9. Losses Faster R-CNN - Losses

When the Faster R-CNN training was complete a frozen inference graph was generated from the latest checkpoint of the Faster R-CNN. The frozen inference graph is the object classifier. The frozen inference graph of dataset ten was used to check that it was able to correctly classify any of the twelve ballet positions in the defined ballet dance positions in the movement dictionary.

3.5.6 Datasets R-CNN Faster-R-CNN-Inception-v2

Dataset nine was comprised of the images used in dataset seven with some images removed and some new images added. Photographs were taken of another volunteer who posed in the same positions as the other volunteers. This volunteer signed the consent form. Ten categories were created for dataset nine's defined ballet dance positions in the movement dictionary. These positions comprised of some positions of the feet, some levels of the feet, and some positions of the arms. The feet were classified as first, second and fifth position with the right foot in front. In the dataset, the positions of the feet were labelled as '1st', '2nd' and '5th_r'. The position of the knees were classified as straight or demi plié. In the dataset the knees are labelled as 'straight_legs' and 'demi_plie'. The levels of the feet were classified as en pointe. In the dataset, the level was labelled as 'en_pointe'. The positions of the arms were classified as bras bas, first, second and fifth. In the dataset the arms are labelled as 'bras_bas', '1st_arms', '2nd_arms' and '5th_arms'. It is evident from the

classifications it would not be possible to create a BMN script from the created object classifier. There are many positions that were not classified or labelled; this was due to the time restrictions in place. The images were labelled using labelImg.py. The training the Faster R-CNN Inception-v2 on dataset nine was used as a personal learning mechanism, due to lack of familiarity with the Faster R-CNN algorithm.

Dataset ten was comprised of images used in dataset nine with some of the images removed. New images were gathered from Google Images and frames from videos were also added. Overfitting can be the result of utilising recurring elements. All of the images from dataset nine were relabelled and the new images were labelled using labelImg.py. Dataset ten contained one thousand and eleven images in total, seven hundred and three training images and three hundred and eight test images. Dataset ten was split 70% training and 30% testing. The selected images for dataset ten, which contained twelve defined ballet dance positions in the movement dictionary, were a more appropriate division of steps to be used in classification to create a BMN script.

3.5.7 Videoing Students in Dublin

Students from the National Performing Arts School (NPAS), in the Lir Studio, Dublin 2, agreed to being recorded while dancing in their class. Two students participated in the class on the day of videoing. As the GDPR Data Protection legislation came law on 25 May 2018, it was essential to explain clearly to both of the student participants and the students' parents to ensure that they understood that the video was for research purposes. The video would be saved on an external USB device and would only be used when testing the newly trained Neural Network. The concept of the dissertation was explained to both students and their parents. As both students were under eighteen years of age, both the students and their parent/guardian signed the permission slip agreeing to participate in the videoing of their dancing. Both of the students had just completed their Grade 5 Royal Academy of Dancing (RAD) ballet exam under the supervision of their ballet teacher. The students danced the Centre practice enchaînement from Grade 5. The Centre practice enchaînement consists of battement tendu devant, battement frappé à la seconde, chassé à la seconde and pirouette en dedans of the legs and feet. This enchaînement encompasses arm

positions of bras bas, first position, second position and third position. The enchaînement is danced on both right and left sides of the body.

3.5.8 Testing the Faster R-CNN Inception-v2 on the recoded video

Two students took part in the recording session. In order to test the image classification, it was essential to extract the image of one of the students. If two students danced, the output of the video log in JavaScript Object Notation JSON file format could return conflicting results and affect the final BMN output score. It was possible to extract the image of one of the dancers using iMove software. It was not possible to change the aspect ratio of the video. However, using iMove, it was possible to zoom and focus on the dancer on the left hand side of the video screen. Initially, the video image was rotated ninety degrees to the left to focus on the image of the dancer on the left and crop the image of the dancer on the right hand side. The video was then saved in an MP4 file format. The video was re-edited and rotated back ninety degrees to the right so that the dancer was in the original vertical position.

3.5.9 Checking that the Faster R-CNN can identify steps from video

Once the video contained the image of only a single dancer it was possible to test the video using the 'Object_detection_video_transcript.py' file. This Python script uses the frozen inference graph which is created by the output of the Faster R-CNN retraining process. The number of objects to be detected is set to twelve, the same number as the defined ballet dance position dictionary created for dataset ten. NUM_CLASSES = 5 is amended to NUM_CLASSES = 12 to the number of classes of the objects to be detected in the input video. The Python script processes the video in MP4 file format file frame by frame. Each frame is checked to see if any of the twelve dance positions in the defined ballet dance positions in the movement dictionary is present in the video. If the inference graph's object detection is able to classify any of the movements it will highlight any of the twelve defined ballet dance positions in the movement dictionary that are present in the frame. The script notes the screen coordinates of the defined ballet dance positions in the movement

dictionary in the frame and outlines these in a coloured bounding box along with its probability score.

The 'Object_detection_video_transcript.py' Python script was amended to create a transcript of the frame-by-frame classification. The threshold was set to 0.9 this was to ensure that the returned classifications were only within the 90% confidence interval. In this transcript the script noted the frame number, the probability score, the object id number, the object name that was created in the 'labelmap.bptxt' and the coordinates of the bounding boxes. This data was placed in the defined ballet dance positions in the movement dictionary structure. This dictionary format gathered information which was then exported into a JSON file format called 'transcript_of_positions.json' for ease of use at a later stage.

3.6 Benesh Notation Editor (BNE)

A copy of Benesh Notation Editor (BNE) was provided by the Royal Academy of Dance. The BNE lite software is compatible solely with Windows operating systems. Having downloading bne2_lite.exe it was necessary to import the 'BNE fonts'. Without these fonts, the BNE lite software would not function. It is necessary to have an understanding of BMN in order to use the BNE lite software successfully. In the software interface the menu is divided into feet, arms and knees. There are also movement lines to show the quality and types of movement. The user must place the symbols representing the body parts in the correct place on the BMN stave, to create a comprehensible BMN output score.

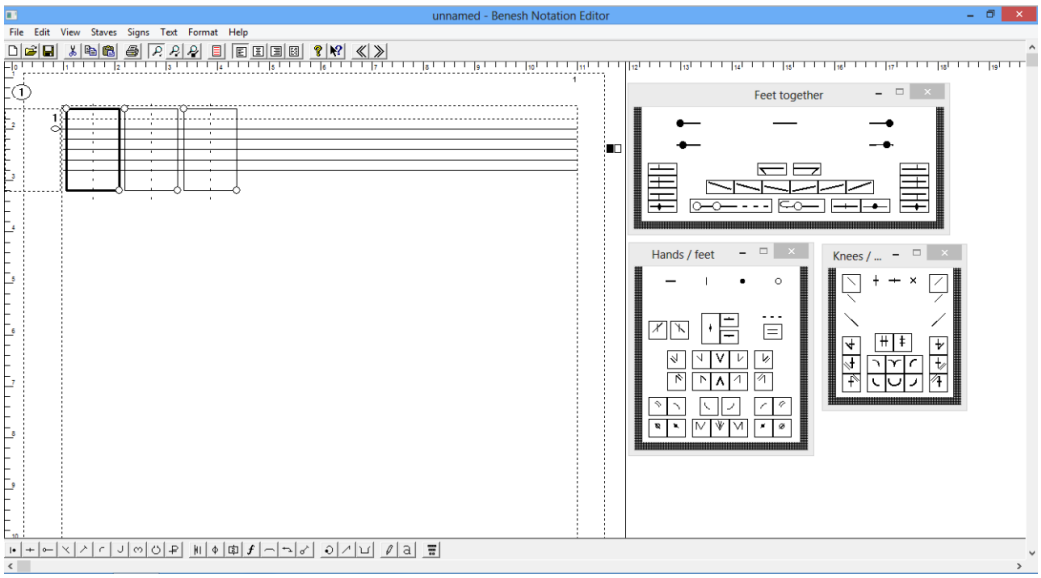

Figure 10. BNE lite Software Interface

The BMN images have been created using the BMN lite software. The feet in first position were created at the three different levels. This was done for the feet: first position, second position, fifth with the right foot in front and the left foot in front. The knees: straight and demi plié, and the arms: bras bas, first and second position were drawn in separate frames see Fig. This script was then exported to a png file format. The png file was edited in Adobe Photoshop.

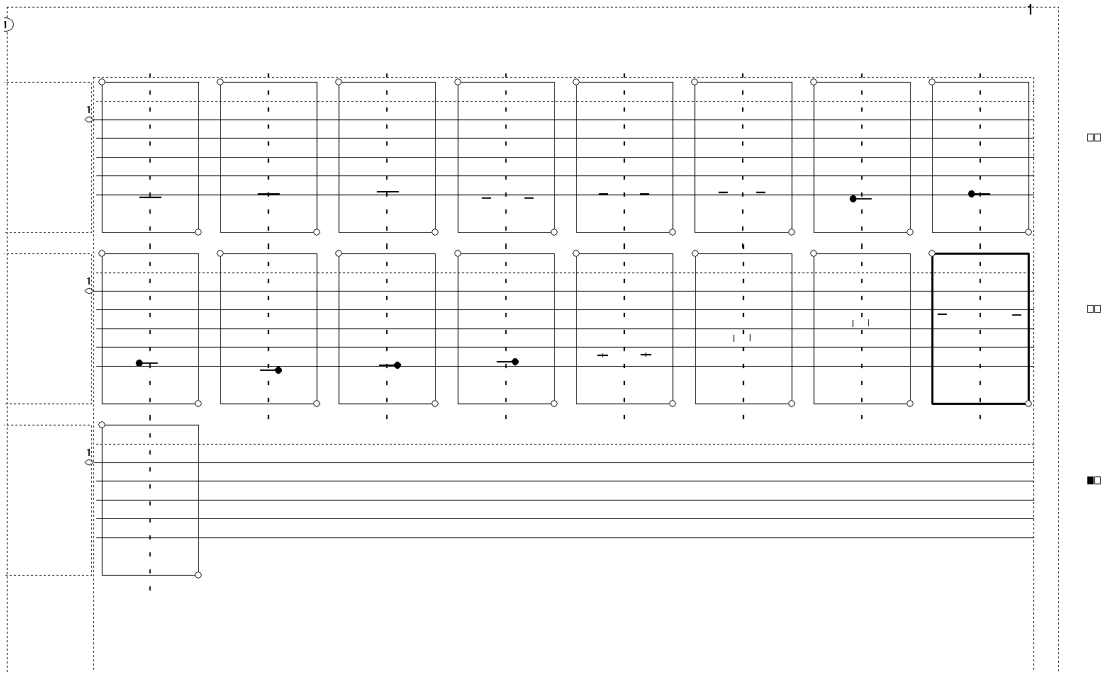

Figure 11. The defined ballet dance positions in the movement dictionary in BMN lite Software

The background was extracted for producing the BMN output score. Later when reproducing the notation score, the legs, knees and feet could be layered on top of each other to reproduce a full picture of the movement that was identified in the video.

It was possible to create the BMN script through the use of two Python scripts: 'Object_detection_video_transcript.py' and 'transcript_to_BMN-final.py'. The former Python script observes the video, creates a dictionary structure of the identified defined ballet dance positions in the movement dictionary in a JSON file format and writes this dictionary to the 'transcript_of_positions.json' file. The latter Python script utilises the data gathered in the 'transcript_of_positions.json'. It processes this data, which is then used to call the separate images and layer them on top of each other to produce the BMN for each identified step from the defined ballet dance position in the movement dictionary. It then organises them sequentially to produce the final BMN output score.

The 'transcript_to_BMN.py' script processes the 'transcript_of_positions.json' reads the JSON log and converts it to BMN. This is done through the use of a def function. The def function counts the number of occurrences of the level of the feet and the positions of the feet knees and arms. The most frequently counted position of each returns the equivalent file. To increase the processing speed in returning the BMN png file, the script does not use if statements. This is achieved through the naming process of each of the images.

Initially, the returned BMN output score was carried out utilising the matplotlib library. However, the BMN output score's placement of the images was returned in split and separated cells. The appearance of the returned BMN output score was incorrect. On further research the pygame library was located. This was able to return an acceptable image. The returned BMN output score had continuous lines.

4 Research Findings and Recommendations

4.1 CNN Inception-v3

The CNN Inception-v3 was able to identify some of the defined ballet dance positions in the movement dictionary. However, it was not accurate enough or specific enough and it did not carry out what was required. If an image contained a dancer carrying out more than one of the defined ballet dance positions in the movement dictionary it could identify some of the positions but not all would be listed in the returned results.

The percentage result was always spilt among five positions. Therefore the percentage of probability was always divided among the five. With the image below where the dancer is in second position and doll like arms, the return result is a mix between second position and fifth position of the feet with only a 30% certainty that the position was second. This was not a strong enough returned result. The ability to label each image to distinguish the position prior to training produced a better return.

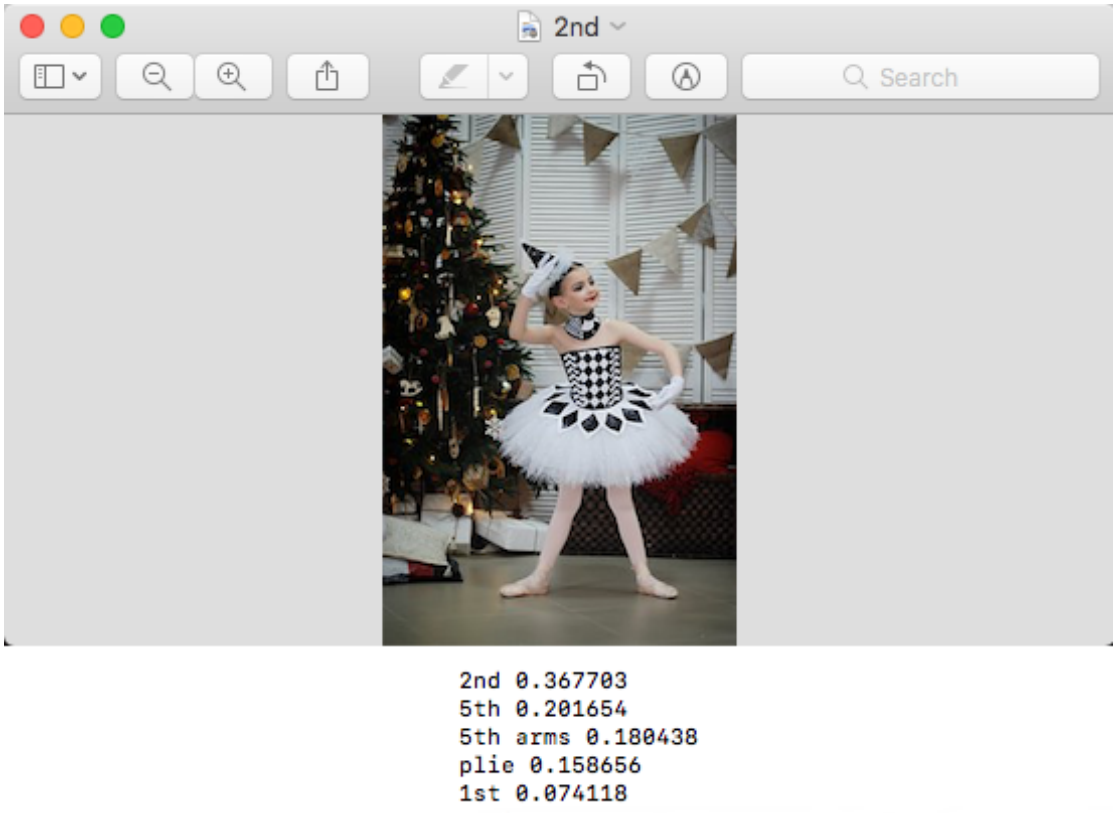

Figure 12. Results CNN Inception-v3.. Static image


```
5th arms 0.231662
plie 0.19417
2nd arms 0.170107
2nd 0.161483
1st 0.100129
```

Figure 13. Results CNN Inception-v3. Static image

4.2 Faster R-CNN Inception-v2

As it did not run its full training course, the Faster R-CNN was not fully trained and as a result, it was not as accurate as it could have been. If its losses had dropped consistently below 0.05 the Faster R-CNN's accuracy should have increased. This lack of accuracy was caused by time restrictions as well as the lack of GPU processing power available on the Mac OS. However, though the Faster R-CNN Inception-v2 algorithm was trained with only 77,436 iterations, it was still able to classify the majority of the positions in the Centre practice enchanîment danced in the dancer1.mp4.

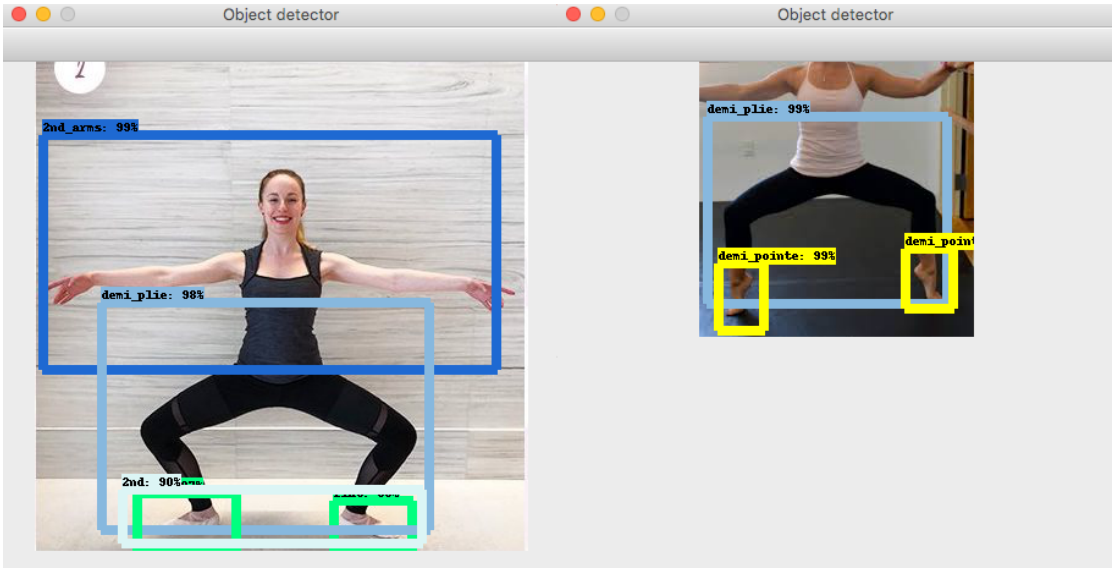

Figure 14. Results Faster R-CNN Inception-v2 - Second Time Running – static images. All positions correctly classified – Right image: Level, flat. Feet – second position. Knees – demi plie. Arms – second position. Left image: Level, demi pointe. Feet – second position. Knees – demi plié

When the Faster R-CNN algorithm was tested on the dancer1.mp4 file with the confidence level set at 0.9 (90%), the algorithm had the ability to correctly identify both levels: flat and demi pointe, feet in second position, knees both straight and in a demi plié and arms in bras bas, first and second position and on occasion fifth with the right foot in front. The algorithm misclassified level pointe and demi pointe. The algorithm is not yet efficient at distinguishing between demi-point and pointe.

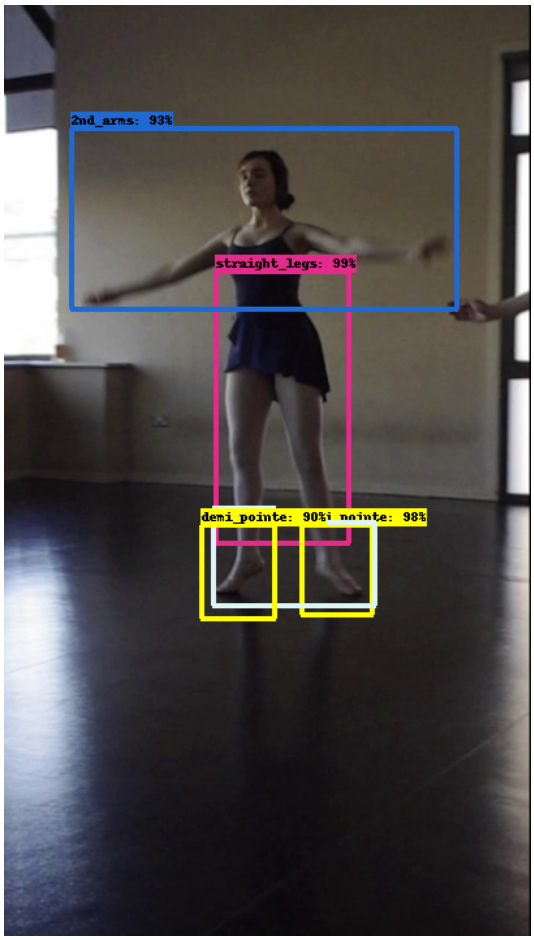

Figure 15. Results Faster R-CNN Inception-v2 – second time running on video All positions correctly classified Level, demi pointe. Feet – second position. Knees – straight. Arms – second position

In dataset ten, there were two hundred and thirty nine instances of demi pointe labelled and one hundred and forty five instances of pointe. In ballet, when the foot is in a flat position it is very different to when in demi pointe or pointe. But there is a smaller difference between the shape of the foot when in demi pointe than in pointe. The foot's ankle and its arch are quite similar in both positions. The small distinguishing feature is the difference between the tip of the toe and the ball of the foot on the floor. However, if the foot is en pointe, the shape and size of the platform on the floor is similar due the shape of the pointe shoe. Perhaps to classify the difference, it would be better to label pointe, with the foot fully extended and the tip of the toe on the floor, and en pointe separately. When en pointe the dancer would be wearing pointe shoes and standing on the tips of the toe. This could help differentiate between demi pointe and pointe in classification.

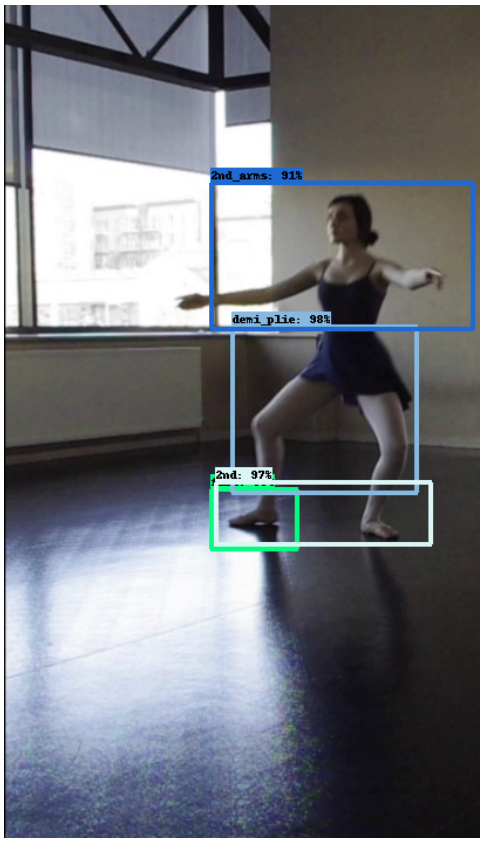

Figure 16. Results Faster R-CNN Inception-v2 – second time running on video. All positions correctly classified Level, flat. Feet – second position. Knees – demi plié. Arms – second position

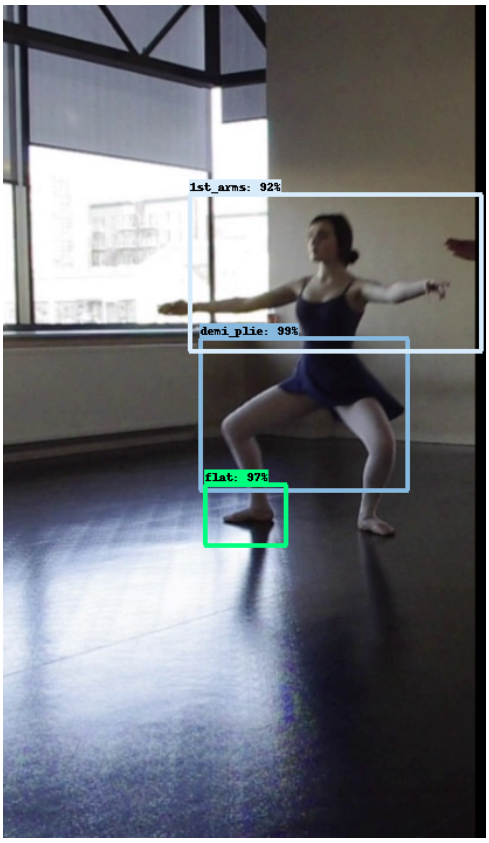

Figure 17. Results Faster R-CNN Inception-v2 – second time running on video. Incorrect classified positions: Arms – should be second position. Correct classified positions: Level, flat. Feet – second position. Knees – demi plié

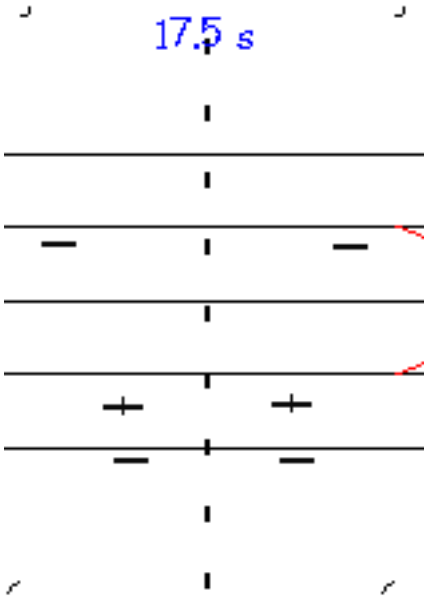

Figure 18. BMN output score of Fig 16

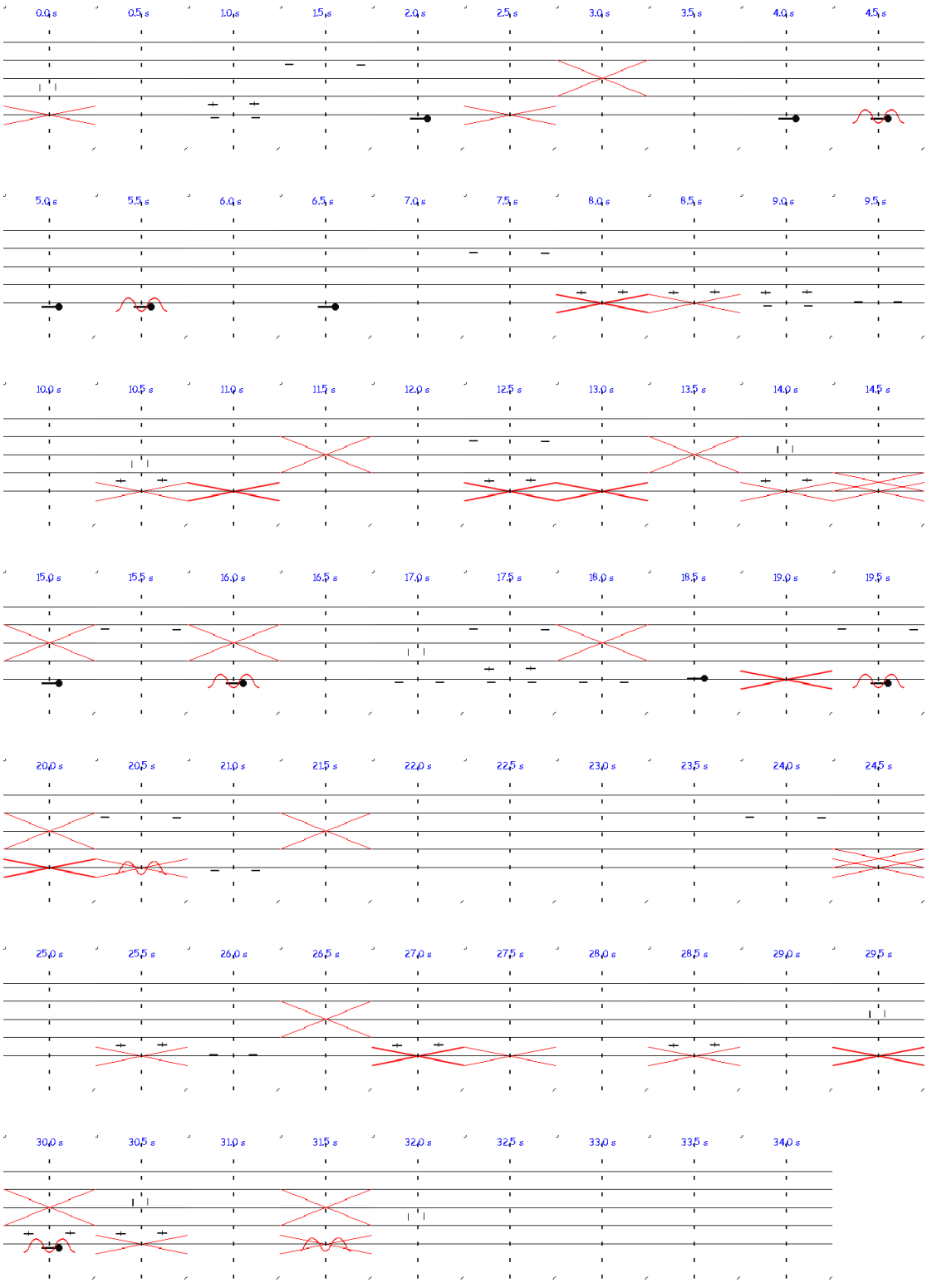

Figure 19. BMN output score of dancer1.mp4

In the dancer1.mp4, on one occasion, the arms were classified as first position when they were actually in second position. The reason for this occurring is not clear. Perhaps an imbalance in the number of labelled arms in first position was not equal to

the number of labelled arms in second position in the data set. In the test dataset there were more arms in first position, 46, than arms in second position, of which there were 34. The training dataset contained a total of 144 arms in first position and 136 arms in second position. Further research is required to identify the reason for the misclassification.

To gather good quality ballet dance positions and build a quality database of images would require the submission of suitable images from a reputable dance organisation. Accompanying permission would be required to comply with current GDPR legislation. A larger dictionary would then be created allowing further development of the database. Movement could be incorporated into the Faster R-CNN algorithm classification to help capture the transition between ballet steps and positions.

5 DISCUSSION

5.1 PROCESSING POWER REQUIRED AS WELL AS TRAINING DURATION

The images used for the datasets came from various sources. The images taken from Google Images were approximately 25kb with dimensions of 639 x 639. The images taken from the converted YouTube videos were approximately 209 KB with dimensions of 1280 x 720. The images taken of the participants were approximately 717kb with dimensions of 3072 x 2048. For accurate training and results, the quality of the image is important when training a Faster R-CNN. The images taken could have been reduced to lessen the required processing power to run the Faster R-CNN Inception-v2 algorithm. Time did not permit this reduction.

A 'resize.py' Python script could have reduced the amount of processing power required to run the Faster R-CNN Inception –v2 algorithm. This python script would reduce the size of the images contained in the same folder as the script to 1024 x 768. This Python script should have been run on the images prior to labelling the images. This also could have also reduced the duration of the training time.

5.2 LABELIMG

When labelling the images on the first time running, some images were over-labelled. Arms and feet positions were unnecessarily labelled. These were not part of the defined ballet dance positions in the movement dictionary of steps. This over-labelling had been done in the hope of speeding up the labelling process for later use. However, when too many images were labelled, the xml files for each image did not match up with the labels set in the CNN. As a result, some images were misclassified on the first time running which resulted in incorrect training resulting in additional unnecessary tasks.

On the second time running, when retraining the Faster R-CNN, the images were relabelled in a different manner. Each level, flat, demi pointe and pointe had its own outlining box. Each individual foot was labelled, to allow for one foot flat on the floor while the other was extended en pointe. This box for defining the levels

encompassed the anklebone down to below the foot and also captured some of the floor. The feet positions were labelled in a single box, encompassing above the ankle and the entire foot. The knees were labelled as a unit in a box from just below the waist to the ankles. The arms were labelled as a unit, encompassing the head to the tips of the fingers. As the BMN output score image production was at an early stage of development, it was returning static movements only that occur on the beat of the bar.

The 'transfer_to_BMN.py' Python script omitted movements that occurred throughout the bar of music or on an uneven beat.

A method of removing noise and uncertainty of the classification of the defined ballet dance positions in the movement dictionary was required to avoid drawing every single movement per second whilst still catching the movements that occur on the uneven beat within a bar of music. The use of Robust Kalman algorithm could help in the reduction of uncertainties and also could notate movements that occur on the uneven beat of the bar.

The empty frames in the classification are the result of the Python script's inability to see the changes in the movement of the dancer. It interprets the information as the dancer still being in the same position. Also, not all of the potential dancer positions have been identified as the twelve defined ballet dance positions in the movement dictionary.

5.3 Time Constraints

The process of gathering data took a significant amount of time. The Python script 'video_to_frame.py' (Weaver, 2018) enabled gathering of more images, which in turn helped to capture the dancers in various positions. Scrolling through the frames of each video was time consuming. When more than one thousand images of dancers in various ballet dance positions were gathered, it was possible to categorise the images. Due to the time constraints of this research, the availability of pre-processed dataset would have reduced the amount of time spent on sourcing images.

5.4 CNN DATA GATHERING

Classifying and labelling the images took a significant amount of time. When working with the CNN, the task of data classification was an exercise of trial and error as there was no clearly defined method of classifying the data. Examples outlined are generally about classifying a species of flower or a breed of dogs. The specific species were grouped into folders and were processed from that. These examples were well documented. For the defined ballet dance positions in the movement dictionary it was difficult to know whether it was appropriate to use the same image in different classifications. It was difficult to decide if a dancer in a demi plié in first position should it be labelled solely as demi plié, solely as first position or in both demi plié and first positions folders.

The intention was to use OpenCV to detect the object within the image. However, without the use of a GPU, errors would occur. However, it was possible to label the images using labelImg to retrain a Faster R-CNN Inception–v2. LabelImg allowed for precision when labelling, but this took a significant amount of time. It was possible to get several classifications in a single image. It was possible to label the dancers' level and the position of the feet within a single image. The knees including the dancers' arms were also labelled in this image. In dataset ten, 3,757 labels were assigned to the 1,033 images.

If attempting this research again, greater focus would be best placed on object detection rather than image classification.

5.5 FASTER R-CNN

The script reduces the size of the input images in the Faster R-CNN. On reflection, it would be better practice to have all images the same size before firstly labelling and then secondly inputting into the Faster R-CNN. This would reduce both the storage space and processing power required to label the images and to process them in the Faster R-CNN.

5.6 Limitations of the Study

The current BMN output script captures the static positions that are occurring every twelfth frame of the dancer1.mp4 video, every half second. This is not in keeping with the timing of the music to which the student is dancing. With more time, it would be possible to create a more musically accurate BMN output script. This would create be able to create a more musically accurate BMN output script. This could be achieved by tailoring the 'Object_detection_video_transcript.py' to each piece of music to which the students would be dancing.

The types of movement are not currently being represented on the BMN output script. When the musical timing has been rectified and the BMN output script represents the dancers musically, it may be possible to represent the types of movement which are occurring when the dancers are moving. Further research would be required in order to develop this facility.

The Faster R-CNN would need to be trained to accurately identify types of movement, for example, a battement tendu or a chassé which has a sliding action, a battement glissé or a piqué which has a sharp staccato action.

At the time of writing, the 'Object_detection_video_transcript.py' script is only functioning one dancer. If there were two dancers in the video frame the results would be doubled. If both dancers were enacting exactly the same movements at the exact same time, it should not interfere with the results that produce the BMN output script. However, if the two dancers were not enacting the same movement the 'transcript_to_BMN.py' script would utilise the dancer's movement that has the highest probability of being the identified movement from the defined ballet dance positions in the movement dictionary.

Due to the lack of processing power and time restrictions, the Faster R-CNN was not trained to its full capacity. A more powerful computer may have permitted faster training with superior results.

6 CONCLUSIONS AND RECOMMENDATIONS

The purpose of this research was to train a Convolution Neural Network (CNN) to see if it could identify specific dance action/steps in a defined ballet dance positions movement dictionary. Subsequently it was intended to attempt to display the identified dance action/steps in Benesh Movement Notation (BMN).

A CNN and Faster R-CNN were retrained to test the capability of a neural network to identify the defined ballet dance positions in the movement dictionary. Both still images and static images from video were utilised in the retraining process. The CNN had the ability to classify some of the defined positions, but the Faster R-CNN's process of labelling made the returned classifications more accurate. A larger dataset that contained a more evenly balanced number of the defined ballet dance positions in the movement dictionary would have returned greater Faster R-CNN accuracy. By training, both CNN and Faster R-CNN are capable of dance action/step classification.

Labelling the images within the Faster R-CNN, as opposed to placing the images in the designated folder in the case of the CNN, created a more accurate training model, as the specific body parts relating to the movement were isolated by the labelling process. The CNN model operated as image classification whereas the Faster R-CNN operated as object detection. Consistency in labelling was an essential part of identifying the dance action/steps in the defined ballet dance positions in the movement dictionary, as inconsistent labelling impacted negatively on the classification results.

Previous researchers have indicated that training a CNN in order to classify videos, even with high specification GPUs, can take a number of weeks. Limited resources in terms of high specification computers, and time, did not allow for CNN training of videos.

As a result a Faster R-CNN was trained in order to see if training highly labelled images could identify ballet steps/action in the defined ballet dance positions in the movement dictionary. The above-mentioned limitations did not allow for complete training of the Faster R-CNN. However, when the Faster R-CNN was tested on the

training that did occur, and only images that fell within the 90% confidence interval were acknowledged, as the threshold was set to 0.9, the Faster R-CNN was able return a good number of positive results. The Faster R-CNN had difficulty in differentiating between demi pointe and pointe. With more images and the addition of a new defined ballet dance position in the movement dictionary, for example, en pointe, this differentiating problem could possibly be overcome.

With the aim of reducing misclassifying when labelling the images for training the Faster R-CNN, a more efficient method of labelling could be achieved by filtering the images by the most distinguishing feature, for example, focusing on the positions of the feet. These images could then be placed in a specified folder. When testing a video, the Robust Kalman's algorithm could be used to reduce random noise.

The processing time of the Faster R-CNN may be reduced if the image size is standardised. This could be achieved using a 'resize.py' Python script. Improved accuracy in the training process may be achieved by increasing the amount of the available data, as extra classifiable positions in the defined ballet dance positions in the movement dictionary would be available.

A concise label map should be created to help with the labelling of the images. This would help keep account of the labelling process and indicate what was efficient and what required updating. If dance action/steps were added to the defined ballet dance positions in the movement dictionary, a record of what worked and what did not work would be available. Additional data may also help with the classification process of the transition between the defined ballet dance positions in the movement dictionary. However, more data would also require a more powerful processor with access to a high specification GPU.

Training of the Faster R-CNN to identify correct ballet or sport technique could help to improve proficiency with concomitant injury reduction. The training of the Faster R-CNN demonstrates that it is possible to correctly classify dance action/steps and test the results on a recorded video of ballet students.

At half-second intervals, it is possible to return a basic BMN output score which identifies the dance action/steps which are defined in the ballet dance positions in the movement dictionary. A method to improve the returned BMN output score notation would require further investigation. This would need to include musical elements such as the music's time signature and in addition the dance action/steps carried out as they occur musically and as choreographically intended. This would require development of a method of capturing musical elements. In addition, the ability to record multiple dancers would increase the usability of a trained Faster R-CNN.

The ability to classify movements and the precise path of a limb correctly, could be adapted to other areas such as physical and sports therapy and rehabilitation. It could be used to monitor and guide rehabilitation programmes and could contain predetermined exercises. Inclusion of a facility to monitor patient progress between medical visits could reduce rehabilitation programme costs.

7 APPENDICES

7.1 APPENDIX A

May 2018.

Re: Permission to observe your group's lesson and video an enchaînement while participants are dancing.

Dear Participant,

I am both an RAD ballet and ISTD modern dance teacher. My primary degree is in dance education. Currently, I am a student at Carlow Institute of Technology, completing an MSc in Data Science and am undertaking research for my dissertation. My dissertation topic is the investigation of the use of machine learning algorithms with image and video classification. The intention is to create a model, which identifies a specific dance action/step in ballet. This model needs to be tested on a recorded performance by ballet dancers. The research focus is on using a Machine Learning algorithm to identify the specific dance action/step and convert the dance action/step to Benesh Movement Notation (BMN) software.

I am seeking your permission to attend your ballet lesson, with a view to observing and recording an enchaînement. This video will be used to convert the enchaînement to BMN software.

I am attaching a Consent Form for your attention.

Many thanks.
Yours faithfully,
Grainne Mulvey
Gráinne Mulvey

* Please note that the video will be used solely for this dissertation research.

✂----------

Consent Form

Do you consent to my videoing an enchaînement performed during class?

Yes ☐ No ☐

Signature of Participant _____ Date_____

*If participant under 18 years of age
Signature of participant's Parent/Guardian_____Date_____

7.2 APPENDIX B

June 2018.

Re: Permission to photograph you in various ballet/dance positions.

Dear Participant,

I am a student at Carlow Institute of Technology, completing an MSc in Data Science and am undertaking research for my dissertation. The dissertation topic is the investigation of the use of machine learning algorithms with image and video classification. The intention is to create a model, which identifies a specific dance action/step in ballet. This model needs to be tested on a recorded performance by ballet dancers. The research focus is on using a machine-learning algorithm to identify the specific dance action/step and convert the dance action/step to Benesh Movement Notation (BMN) software.

I am seeking your permission to photograph you in various ballet/dance positions. These photographs will be used to create part of my database for use in the MSc Dissertation.

I am attaching a Consent Form for your attention.

Many thanks.

Yours faithfully,
Grainne Mulvey
Gráinne Mulvey

* Please note that the photograph will be used solely for this dissertation research.

✂----------

Consent Form

Do you consent to your photograph being taken and used as part of my database for use in my MSc Dissertation?

Yes ☐ No ☐

Signature of Participant _____ Date_____

7.3 Appendix C

7.4 Python scripts

7.4.1 video_to_frame.py

```
## from
https://gist.github.com/keithweaver/70df4922fec74ea87405b83840b45d57
'''
Using OpenCV takes a mp4 video and produces a number of images.

Requirements
----
You require OpenCV 3.2 to be installed.

Run
----
Open the main.py and edit the path to the video. Then run:
$ python main.py

Which will produce a folder called data with the images. There will
be 2000+ images for example.mp4.
'''
import cv2
# import numpy as np
import os

# Playing video from file:
cap = cv2.VideoCapture('positions_feet.mp4')

try:
    if not os.path.exists('positions'):
        os.makedirs('positions')
except OSError:
    print('Error: Creating directory of data')

currentFrame = 0
while (True):
    # Capture frame-by-frame
    ret, frame = cap.read()

    # Saves image of the current frame in jpg file
    name = './positions/frame' + str(currentFrame) + '.jpg'
    print('Creating...' + name)
    cv2.imwrite(name, frame)

    # To stop duplicate images
    currentFrame += 1

# When everything done, release the capture
cap.release()
cv2.destroyAllWindows()
```

7.4.2 OBJECT_DETECTION_TRANSCRIPT.PY

```
######## Video Object Detection Using Tensorflow-trained
Classifier #########
# Author: Evan Juras
# Date: 1/16/18
# Description:
# This program uses a TensorFlow-trained classifier to perform
object detection.
# It loads the classifier uses it to perform object detection
on a video.
# It draws boxes and scores around the objects of interest in
each frame
# of the video.

## Some of the code is copied from Google's example at
##
https://github.com/tensorflow/models/blob/master/research/obje
ct_detection/object_detection_tutorial.ipynb

## and some is copied from Dat Tran's example at
##
https://github.com/datitran/object_detector_app/blob/master/ob
ject_detection_app.py
## changed it to make it more understandable to me.

#### Edited - while processing the dancer1.pm4 file a
transcript is created.
### The file created is 'transcript_of_positions.json' it is
in a JSON file format.
### This script writes out all the identified steps according
tot he threshold set

# Import packages
import os
import cv2
import numpy as np
import tensorflow as tf
import sys

### added - to export the gathered/identified steps data
import json
###

# This is needed since the notebook is stored in the
object_detection folder.
sys.path.append("..")

# Import utilites
from utils import label_map_util
from utils import visualization_utils as vis_util

# Name of the directory containing the object detection module
```

```
we're using
MODEL_NAME = 'inference_graph'
VIDEO_NAME = './dancer1.mp4'

### added - setting the confidence level of the detected
movements - aka displayed boxes
THRESHOLD = 0.9
###

# Get path to current working directory
CWD_PATH = os.getcwd()

# Path to frozen detection graph .pb file, which contains the
model that is used
# for object detection.
PATH_TO_CKPT =
os.path.join(CWD_PATH,MODEL_NAME,'frozen_inference_graph.pb')

# Path to label map file
PATH_TO_LABELS =
os.path.join(CWD_PATH,'training','labelmap.pbtxt')

# Path to video
PATH_TO_VIDEO = os.path.join(CWD_PATH,VIDEO_NAME)

#Amended to cater for the defined ballet dance positions in
the movement dictionary
# Number of classes the object detector can identify
NUM_CLASSES = 12

# Load the label map.
# use internal utility functions
label_map = label_map_util.load_labelmap(PATH_TO_LABELS)
categories =
label_map_util.convert_label_map_to_categories(label_map,
max_num_classes=NUM_CLASSES, use_display_name=True)
category_index =
label_map_util.create_category_index(categories)

### added - converts the dictionary item {1: { id: 1 name:
'straight'}} to 1: straight etc.
position_names = {}
for pi in category_index:

position_names[category_index[pi]['id']]=category_index[pi]['n
ame']
###

# Load the Tensorflow model into memory.
detection_graph = tf.Graph()
with detection_graph.as_default():
    od_graph_def = tf.GraphDef()
    with tf.gfile.GFile(PATH_TO_CKPT, 'rb') as fid:
```

```
        serialized_graph = fid.read()
        od_graph_def.ParseFromString(serialized_graph)
        tf.import_graph_def(od_graph_def, name='')

    sess = tf.Session(graph=detection_graph)

# Define input and output tensors (i.e. data) for the object
detection classifier

# Input tensor is the image
image_tensor =
detection_graph.get_tensor_by_name('image_tensor:0')

# Output tensors are the detection boxes, scores, and classes
# Each box represents a part of the image where a particular
object was detected
detection_boxes =
detection_graph.get_tensor_by_name('detection_boxes:0')

# Each score represents level of confidence for each of the
objects.
# The score is shown on the result image, together with the
class label.
detection_scores =
detection_graph.get_tensor_by_name('detection_scores:0')
detection_classes =
detection_graph.get_tensor_by_name('detection_classes:0')

# Number of objects detected
num_detections =
detection_graph.get_tensor_by_name('num_detections:0')

# Open video file
video = cv2.VideoCapture(PATH_TO_VIDEO)

### added - opens up file to write to - csv
frame_index = 0
transcript_of_positions = {}

while(video.isOpened()):
    ### added ---- to keep track of the frame number
    frame_index += 1
    transcript_of_positions[frame_index]={}
    # Acquire frame and expand frame dimensions to have shape:
[1, None, None, 3]
    # i.e. a single-column array, where each item in the
column has the pixel RGB value
    ret, frame = video.read()
    frame_expanded = np.expand_dims(frame, axis=0)
    try:
        # Perform the actual detection by running the model
with the image as input
        (boxes, scores, classes, num) = sess.run(
```

```
            [detection_boxes, detection_scores,
detection_classes, num_detections],
            #returns 300 possible regions
            feed_dict={image_tensor: frame_expanded})
    except:
        break
# loop through all 300 possible detected region in this(each)
frame keeping track of what position in the 300 its on
    idx=0
    # loop through all 300 scores i is the value of the score
on position idx
    for i in scores[0]:
        if i is None:
            break
        # if the confidence score is above given threshold
stated above then process it
        elif i>= THRESHOLD:
            transcript_of_positions[frame_index][idx] = {
                'score':float(i*100),
                'position_id':int(classes[0][idx]),

'position_name':position_names[int(classes[0][idx])],

'area':{'x1':int(boxes[0][idx][0]*1000),'y1':int(boxes[0][idx]
[1]*1000),'x2':int(boxes[0][idx][2]*1000),'y2':int(boxes[0][id
x][3]*1000)}
                 }
            #print(transcript_of_positions)
        #else, breaks out of loop as scores are sorted in
descending order onto next frame
        else:
            break
        # increment the counter as looping through the scores
        idx+=1
    #########transcript.writerow([frame_index,  detection_boxes,
detection_classes, detection_scores])
    # Draws the results of the detection (aka 'visulaise the
results')
    vis_util.visualize_boxes_and_labels_on_image_array(
        frame,
        np.squeeze(boxes),
        np.squeeze(classes).astype(np.int32),
        np.squeeze(scores),
        category_index,
        use_normalized_coordinates=True,
        line_thickness=4,
        min_score_thresh=THRESHOLD)

    # All the results have been drawn on the frame, so it's
time to display it.
    cv2.imshow('Object detector', frame)


    if frame_index % 10 == 0:
```

```
        ## edited
        jsond = json.dumps(transcript_of_positions,
separators= (',:'),indent=2)
        f = open('transcript_of_positions.json', 'w')
        f.write(jsond)
        f.close()

    # Press 'q' to quit
    if cv2.waitKey(1) == ord('q'):
        break

## added writing out the data to the JSON file
transcript_of_positions
jsond = json.dumps(transcript_of_positions, separators=
(',:'),indent=2)
f = open('transcript_of_positions.json', 'w')
f.write(jsond)
f.close()
####
video.release()
cv2.destroyAllWindows()
```

7.4.3 TRANSCRIPT_TO_BMN.PY

```
import json
import pygame


NUM_CLASSES=12
BMN_DIR='./BMN_Images/'

images={}

def get_most_frequent(position_list, frequency_count_dict):
    max=0
    max_position=0
    counter=0
    for position in frequency_count_dict:
        if position in position_list:
            if (max<frequency_count_dict[position]):
                max=frequency_count_dict[position]
                max_position=position
    return max_position

def get_image(image_name):
    global images
    if images.get(image_name) == None:
        try:

images[image_name]=pygame.transform.scale(pygame.image.load(BM
N_DIR + image_name + '.png'),(140,220))
        # if unsure get the position file that returns an
unsure represnetation
        except:
            print(BMN_DIR + image_name + '.png')

images[image_name]=pygame.transform.scale(pygame.image.load(BM
N_DIR + 'X.png'),(140,220))
    return images[image_name]

json_file = open('transcript_of_positions.json', 'r')
transcript_of_positions = json.loads(json_file.read())

pos_counter={0:0,1:0,2:0,3:0,4:0,5:0,6:0,7:0,8:0,9:0,10:0,11:0
,12:0}
# first cell
pos_no=0


# Define colours
BLACK = (0, 0, 0)
WHITE = (255, 255, 255)
GREEN = (0, 255, 0)
BLUE = (0, 0, 255)
RED = (255, 0, 0)
```

```
# initialise pygame - gets everything started
pygame.init()
# set the display mode, canvas size
screen = pygame.display.set_mode( (1400, 1980) )
# load and set the font
pygame.font.init()
myfont = pygame.font.SysFont('Comic Sans MS', 14)

# clear the canvas
screen.fill(WHITE)
page_no=1
last_feet='_'
last_arms='_'
last_knees='_'
printing=0
for frame in transcript_of_positions:
    frame_no=int(frame)-1

    # in each of the 12 frames, count the number of occurances
of the deteced position
    for box in transcript_of_positions[frame]:

position_id=transcript_of_positions[frame][box]['position_id']
        pos_counter[int(position_id)] += 1
    # process the data when it gets to 12th (when all of the
12 frames have been added up)
    if (frame_no+1)%12 == 0:

        # using the pos_counter (position counter) use the the
most requent feet, level, knees and arms positions
        feet_position=get_most_frequent([4,5,6,7],pos_counter)
        level_position=get_most_frequent([1,2,3],pos_counter)
        knees_position=get_most_frequent([8,9],pos_counter)

arms_position=get_most_frequent( [10,11,12],pos_counter)

        # if no "knees" found/counted
        if knees_position == 0:
            # mark as unsure
            knees_position = 'kneesX'
        # if no "arms" found
        if arms_position == 0:
            # mark as unsure
            arms_position = 'armsX'
        # if no "feet" found/counted
        if feet_position == 0:
            # mark as unsure
            feet_position = 'feetX'

        # draw the layers
        # consider changes in feet and knees -at same time as
depend on each other
```

```
        if (last_feet != str(level_position) + '-' +
str(feet_position)) or (last_knees != str(knees_position)):
            #print the image in the cell (feet and level image
file naming as (without the spaces):  <level_id> -
<feet_position_id> . png
            screen.blit(get_image(str(level_position) + '-' +
str(feet_position)), ((pos_no % 10)*140, (pos_no // 10)*286))
            printing = 1
            last_feet = str(level_position) + '-' +
str(feet_position)

            # knees (demi plie or straight)
            screen.blit(get_image(str(knees_position)),
((pos_no % 10)*140, (pos_no // 10)*286))
            printing = 1
            last_knees = str(knees_position)

        # arms (bras bas, 1st or 2nd)
        if last_arms!=str(arms_position):
            screen.blit(get_image(str(arms_position)),
((pos_no % 10)*140, (pos_no // 10)*286))
            printing = 1
            last_arms = str(arms_position)

        # printing empty lines
        if printing == 0:
            screen.blit(get_image('empty'), ((pos_no % 10)*140,
(pos_no // 10)*286))

        # print the time and add the time stamp
        screen.blit(myfont.render(str(0.5*(frame_no // 12))+'
s', False, BLUE),((pos_no % 10)*140+50, (pos_no // 10)*286))
        printing = 0
        # end of the first cell 0 first 12 frames of the video
        # restart the pos counter for the next 12 video frames
        pos_counter = {0: 0, 1: 0, 2: 0, 3: 0, 4: 0, 5: 0, 6:
0, 7: 0, 8: 0, 9: 0, 10: 0, 11: 0, 12: 0}
        # move on to next cell on the grid / sheet
        pos_no += 1

        if (pos_no%70 == 0):
            # display the image
            pygame.display.flip()
            # save it to a file
            pygame.image.save(screen,'./BMN output score -
Centre Practise - page'+str(page_no)+'.png')
            page_no += 1
            # clear the image again
            screen.fill(WHITE)

# and save also the very last page
# display the image
pygame.display.flip()
```

```
# save it to a file
pygame.image.save(screen,'./BMN output score - Centre Practise
- page'+str(page_no)+'.png')

# close the canvas window
pygame.quit()
```

8 REFERENCES

Abadi, M. et al. (2016). TensorFlow: A system for large-scale machine learning. *Savannah, GA, USA*, 2 November 2016, p.21.

Albawi, S., Mohammed, T.A. and Al-Zawi, S. (2017). Understanding of a convolutional neural network. In: IEEE, pp.1–6. Available from: http://ieeexplore.ieee.org/document/8308186/ [accessed 9 June 2018].

Aloysius, N. and Geetha, M. (2017). A review on deep convolutional neural networks. In: IEEE, pp.0588–0592. Available from: http://ieeexplore.ieee.org/document/8286426/ [accessed 9 June 2018].

Baker, B., Gupta, O., Naik, N. and Raskar, R. (2016). Designing Neural Network Architectures using Reinforcement Learning. *arXiv:1611.02167 [cs]* [online], 7 November 2016. Available from: http://arxiv.org/abs/1611.02167 [accessed 14 July 2018].

Bradski, G., Kaehler, A. and Pisarevsky, V. (2005). Learning-Based Computer Vision with Intel's Open Source Computer Vision Library. *Intel Technology Journal*, 9(2), pp.119–130.

Bradski, G.R. and Kaehler, A. (2011). *Learning OpenCV: computer vision with the OpenCV library*. 1. ed., [Nachdr.]. Beijing: O'Reilly.

Chen, S.-T., Cornelius, C., Martin, J. and Chau, D.H. (2018). Robust Physical Adversarial Attack on Faster R-CNN Object Detector. *arXiv:1804.05810 [cs, stat]* [online], 16 April 2018. Available from: http://arxiv.org/abs/1804.05810 [accessed 28 July 2018].

COCO - Common Objects in Context. Available from: http://cocodataset.org/#home [accessed 20 July 2018].

Data Protection Commissioner. (2018). The GDPR and You Preparign for 2018. , 2018. Available from: https://www.dataprotection.ie/docimages/documents/The%20GDPR%20and%20You.pdf [accessed 16 July 2018].

Downloader, Y.M. *Youtube Multi Downloader Online Free* [online]. *Youtube Multi Downloader* [online]. Available from: http://youtubemultidownloader.com [accessed 11 July 2018].

EU General Data Protection Regulation. (2016). , 2016. Available from: https://www.itgovernance.eu/en-ie/green-papers/eu-gdpr-compliance-guide-ie [accessed 16 July 2018].

Forster, M. (2004). *Music notation system*. Google Patents. Available from: https://www.google.com/patents/US20040139843.

Fukushima, K. (1980). Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics*, 36(4), pp.193–202.

Géron, A. (2017). *Hands-on machine learning with Scikit-Learn and TensorFlow: concepts, tools, and techniques to build intelligent systems*. Sebastopol, California: O'Reilly.

Girshick, R. (2015). Fast R-CNN. *arXiv:1504.08083 [cs]* [online], 30 April 2015. Available from: http://arxiv.org/abs/1504.08083 [accessed 8 July 2018].

Girshick, R. *rbg's home page* [online]. *rossgirshick.info* [online]. Available from: http://www.rossgirshick.info/ [accessed 17 July 2018].

Girshick, R., Donahue, J., Darrell, T. and Malik, J. (2013). Rich feature hierarchies for accurate object detection and semantic segmentation. *arXiv:1311.2524 [cs]* [online], 11 November 2013. Available from: http://arxiv.org/abs/1311.2524 [accessed 14 July 2018].

Goldsborough, P. (2016). A Tour of TensorFlow. *arXiv:1610.01178 [cs]* [online], 1 October 2016. Available from: http://arxiv.org/abs/1610.01178 [accessed 1 June 2018].

Gorelick, L., Blank, M., Shechtman, E., Irani, M. and Basri, R. (2007). Actions as Space-Time Shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(12), pp.2247–2253.

Gorelick, L., Blank, M., Shechtman, E., Irani, M. and Basri, R. (2005). Actions as Space-Time Shapes. , 2005, p.8.

Hahne, F., Huber, W., Gentleman, R. and Falcon, S. (2008). *Bioconductor Case Studies* [online]. New York, NY: Springer New York. Available from: http://link.springer.com/10.1007/978-0-387-77240-0 [accessed 29 May 2018].

hub: A library for transfer learning by reusing parts of TensorFlow models. (2018). tensorflow. Available from: https://github.com/tensorflow/hub [accessed 28 May 2018].

Ide, H. and Kurita, T. (2017). Improvement of learning for CNN with ReLU activation by sparse regularization. In: IEEE, pp.2684–2691. Available from: http://ieeexplore.ieee.org/document/7966185/ [accessed 16 June 2018].

ImageNet. (2016). *ImageNet* [online]. Available from: http://imagenet.stanford.edu/index [accessed 20 July 2018].

ImageNet Ballet Dancer. (2010). *ImageNet* [online]. Available from: http://imagenet.stanford.edu/synset?wnid=n09834699# [accessed 13 July 2018].

ImageNet Ballet Master. (2010). Available from: http://imagenet.stanford.edu/synset?wnid=n09834885 [accessed 13 July 2018].

ImageNet Ballet mistress. (2010). Available from: http://imagenet.stanford.edu/synset?wnid=n09835017 [accessed 13 July 2018].

ImageNet Ballet skirt, tutu. (2010). Available from: http://imagenet.stanford.edu/synset?wnid=n02780815 [accessed 13 July 2018].

Kohavi, R. (1995). foAr AStcucduyraocfyCErsotsims-VatailoidnaatinodnManoddeBloSoetlsetcrtaiopn. , 1995, p.7.

LeCun, Y., Bottou, L., Bengio, Y. and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11), pp.2278–2324.

LeCun, Y., Bengio, Y. and Hinton, G. (2015a). Deep learning. *Nature*, 521(7553), pp.436–444.

LeCun, Y., Bengio, Y. and Hinton, G. (2015b). Deep learning. *Nature*, 521(7553), pp.436–444.

McGuiness-Scott, J. (1983). *Movement Study and Benesh Movement Notation*. 1st edition. London: Oxford University Press.

MIT Media Lab. (2016). *Professor Emeritus Seymour Papert, pioneer of constructionist learning, dies at 88* [online]. *MIT News* [online]. Available from: http://news.mit.edu/2016/seymour-papert-pioneer-of-constructionist-learning-dies-0801 [accessed 12 June 2018].

Mitchell, T.M. (2006). The Discipline of Machine Learning. , July 2006, p.9.

Müller, A.C. and Guido, S. (2016). *Introduction to machine learning with Python: a guide for data scientist*. First edition. Beijing: O'Reilly.

O'Shea, K. and Nash, R. (2015). An Introduction to Convolutional Neural Networks. *arXiv:1511.08458 [cs]* [online], 26 November 2015. Available from: http://arxiv.org/abs/1511.08458 [accessed 30 May 2018].

Owens, J.D., Houston, M., Luebke, D., Green, S., Stone, J.E. and Phillips, J.C. (2008). GPU Computing. *Proceedings of the IEEE*, 96(5), pp.879–899.

Pinterest- BMN images. *Pinterest* [online]. Available from: https://pl.pinterest.com/pin/100697741642195216/ [accessed 28 July 2018].

Ren, S., He, K., Girshick, R. and Sun, J. (2016). Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *arXiv:1506.01497v3 [cs]* [online], 6 January 2016. Available from: http://arxiv.org/abs/1506.01497 [accessed 6 July 2018].

Ren, S. (2016). *Shaoqing Ren* [online]. *http://www.shaoqingren.com* [online]. Available from: http://www.shaoqingren.com/ [accessed 17 July 2018].

Royal Opera House. *YouTube* [online]. Available from: https://www.youtube.com/channel/UCHS5XKgf2FCBF8pZllE_bjw [accessed 11 July 2018].

Ryman, R., Singh, B., Beatty, J.C. and Booth, K.S. (1984). A Computerized Editor of Benesh Movement Notation. *Dance Research Journal*, 16(1), p.27.

Saad, S., De Beul, D., Mahmoudi, S. and Manneback. (2012). An Ontology for video human movement representation based on Benesh notation. In: *2012 International Conference on Multimedia Computing and Systems*. 2012 International Conference on Multimedia Computing and Systems. pp.77–82.

Seising, R. and Tabacchi, M.E. (2013). A very brief history of soft computing: Fuzzy Sets, artificial Neural Networks and Evolutionary Computation. In: *2013 Joint IFSA World Congress and NAFIPS Annual Meeting (IFSA/NAFIPS)*. 2013 Joint IFSA World Congress and NAFIPS Annual Meeting (IFSA/NAFIPS). pp.739–744.

Seo, B., Shin, M., Mo, Y.J. and Kim, J. (2018). Top-down parsing for Neural Network Exchange Format (NNEF) in TensorFlow-based deep learning computation. In: *2018 International Conference on Information Networking (ICOIN)*. 2018 International Conference on Information Networking (ICOIN). pp.522–524.

Shechtman, E. and Irani, M. (2005). Space-time behavior based correlation. In: *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*. 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05). pp.405–412 vol. 1.

Shen, Z., Liu, Z., Li, J., Jiang, Y.-G., Chen, Y. and Xue, X. (2018). DSOD: Learning Deeply Supervised Object Detectors from Scratch. *arXiv:1708.01241 [cs]*, 30 April 2018, pp.1–11.

Sinha, T., Verma, B. and Haidar, A. (2017). Optimization of convolutional neural network parameters for image classification. In: *2017 IEEE Symposium Series on Computational Intelligence (SSCI)*. 2017 IEEE Symposium Series on Computational Intelligence (SSCI). pp.1–7.

Sun, C., Shrivastava, A., Singh, S. and Gupta, A. (2017). Revisiting Unreasonable Effectiveness of Data in Deep Learning Era. *arXiv:1707.02968 [cs]*, 10 July 2017, pp.1–13.

Szegedy, C., Wei Liu, Yangqing Jia, Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V. and Rabinovich, A. (2015). Going deeper with convolutions. In: IEEE, pp.1–9. Available from: http://ieeexplore.ieee.org/document/7298594/ [accessed 29 May 2018].

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J. and Wojna, Z. (2016). Rethinking the Inception Architecture for Computer Vision. In: IEEE, pp.2818–2826. Available from: http://ieeexplore.ieee.org/document/7780677/ [accessed 3 July 2018].

TensorFlow Models: detection_model_zoo.md. (2018). tensorflow. Available from: https://github.com/tensorflow/models [accessed 30 May 2018].

tzutalin, darrenl. (2018). *labelImg: LabelImg is a graphical image annotation tool and label object bounding boxes in images* [online]. Available from: https://github.com/tzutalin/labelImg [accessed 6 July 2018].

tzutalin (darrenl). Available from: https://github.com/tzutalin [accessed 6 July 2018].

Uijlings, J.R.R., van de Sande, K.E.A., Gevers, T. and Smeulders, A.W.M. (2013). Selective Search for Object Recognition. *International Journal of Computer Vision*, 104(2), pp.154–171.

Veloso, M. (2018). *Machine Learning - School of Computer Science - Carnegie Mellon University* [online]. *Machine Learning Department at Carnegie Mellon University* [online]. Available from: https://www.ml.cmu.edu/#machine_learning [accessed 27 May 2018].

Weaver, K. (2018). *Using OpenCV takes a mp4 video and produces a number of images.* [online]. Available from: https://gist.github.com/keithweaver/70df4922fec74ea87405b83840b45d57 [accessed 21 March 2018].

Yang, H., Yuan, C., Xing, J. and Hu, W. (2017). SCNN: Sequential convolutional neural network for human action recognition in videos. In: IEEE, pp.355–359. Available from: http://ieeexplore.ieee.org/document/8296302/ [accessed 9 June 2018].

Zhou, Z.-H. (2017). A brief introduction to weakly supervised learning. *National Science Review*, 5(1), pp.44–53.