Name: Seth Ng Jun Jie

1. CNN Implementation:

A. Basic Architecture:

For the original convolutional neural network's (CNN) architecture as specified in the project spec, the version modelled in this project has a max number of epochs of 250. The accuracy and losses of both the training and validation datasets were recorded over time as shown below.

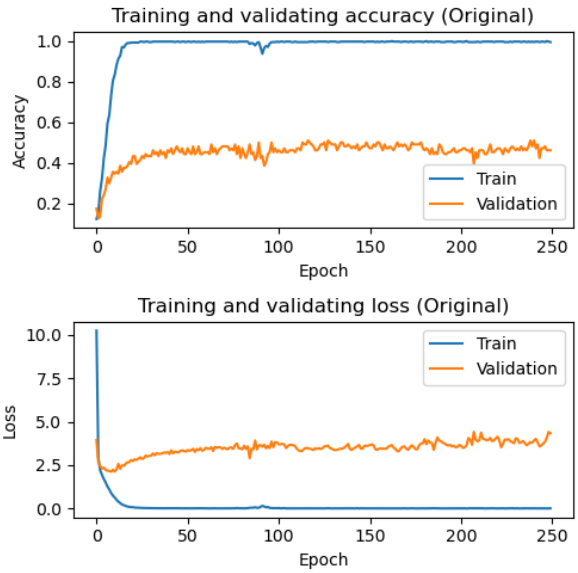

Fig 1: Accuracies and losses of training and validation sets from the Original model.

As shown above, it can be observed that the training set's accuracy rises very early on and quickly reaches close to 100% while its loss decreases just as quickly, reaching close to 0. The validation set's accuracy improves gradually but does not go higher than around 50% and its loss actually go up over time after initially decreasing. This is clearly due to overfitting from the training set. The training set's abnormal steadiness in its rate of increase and the constant and barely-increasing accuracy of the validation set further support this.

B. Regularisation and Data Augmentation:

New aspects were added to the model using regularisation and data augmentation techniques. Our regularisation methods include L1 and L2 regularisers, dropout layers, and early stopping.

An L1 regulariser with lambda = 0.01 was added to the first convolutional layer. L2 regularisers were added to the remaining three, with lambda = {0.001, 0.001, 0.01} in that order. These choices were based off the number of parameters in each convolutional layer. From **Fig 2**, the first one only has 448, compared to remaining three with 2320, 3480, and 5208, therefore, the first layer has less chances to wrongly 'zero-ing' a parameter according to probability so an L1 regulariser is assigned to it. In the other three, 5208 is much larger than 2320 and 3480 so the last layer is assigned a higher lambda value to counter the higher complexity.

Dropout layers are included before each of the two max pooling layers with dropout rates of 0.3. This is to prevent the model's learning from relying too much on the training data, and therefore, overfitting. The dropout rate of 0.3 seemed to give the best performance on the validation set.

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_4 (Conv2D)           (None, 30, 30, 16)        448

 conv2d_5 (Conv2D)           (None, 28, 28, 16)        2320

 dropout (Dropout)           (None, 28, 28, 16)        0

 max_pooling2d_2 (MaxPooling  (None, 14, 14, 16)       0
 2D)

 conv2d_6 (Conv2D)           (None, 12, 12, 24)        3480

 conv2d_7 (Conv2D)           (None, 10, 10, 24)        5208

 dropout_1 (Dropout)         (None, 10, 10, 24)        0

 max_pooling2d_3 (MaxPooling  (None, 5, 5, 24)         0
 2D)

 flatten_1 (Flatten)         (None, 600)               0

 dense_1 (Dense)             (None, 10)                6010

=================================================================
Total params: 17,466
Trainable params: 17,466
Non-trainable params: 0
_________________________________________________________________
```

Fig 2: Summary of the Regularisation model.

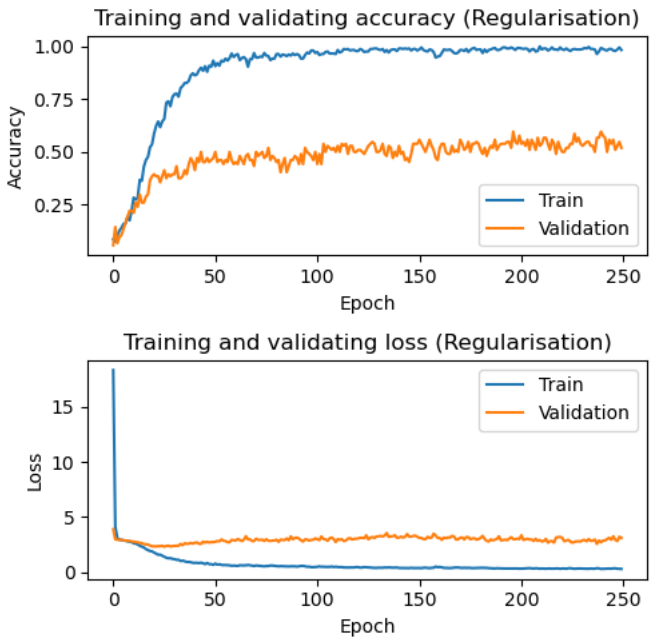

Fig 3: Accuracies and losses of training and validation sets from the Regularisation model.

After looking at the results from adding L1/L2 norms and dropout, early stopping was introduced to the model according to the loss function with a minimum improvement amount of 0.01 and a patience factor of 50 epochs. Looking at **Fig 3**, we see that throughout the 250 epochs the validation set's performance was still able to slowly increase overtime. Therefore, the max number of epochs was changed to 500.

For data augmentation, the transitions considered include horizontal and vertical shifts, zoom, horizontal flips, and brightness. Rotations and vertical flips were not considered as none of the images in the test set have these properties. After adding these and the early stopping the final model was produced.

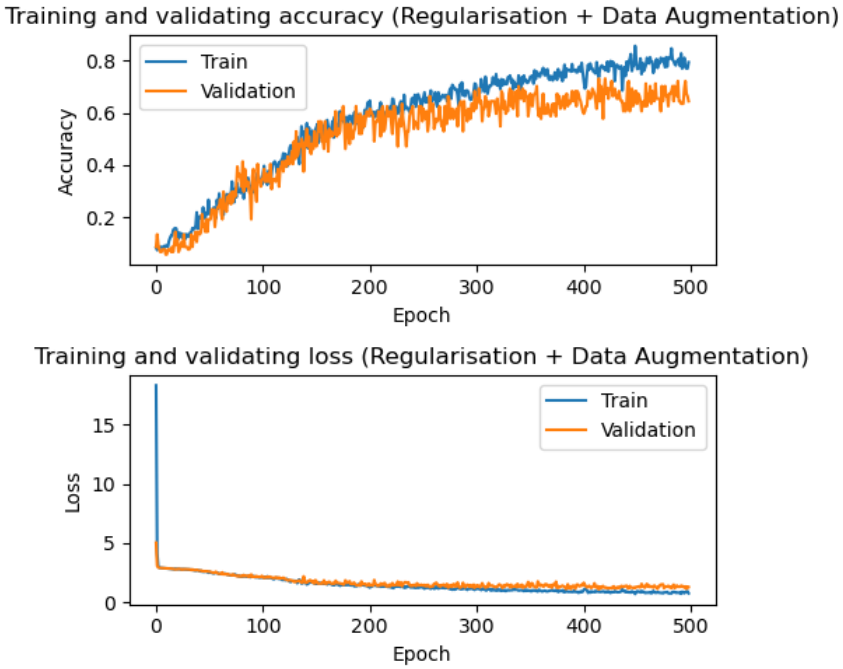

Fig 4: Accuracies and losses of training and validation sets from the Final model.

In **Fig 4**, the performance of the final model with all the modifications mentioned above can be observed. In comparison to the original model in **Fig 1**, the final model has less smooth rises and declines in the training accuracy and loss respectively, indicating that the overfitting problem has been overcome. This is also depicted by the refined performance of the validation set which no longer remains too constant and at certain points even outperforms the training set. The final model's validation loss is much more in line with its training loss compared to the original model's increasing validation loss which implies that the model has successfully learned to generalise.

2. **Error Analysis:**
Applying the final model on the test set an overall classification accuracy of **67.14%** and a loss of 1.613 were obtained. The average accuracies (recalls) obtained from the final model for each of the ten classes are as follows:

	precision	recall	f1-score
bridge	0.67	0.57	0.62
childs	0.57	0.57	0.57
downwarddog	0.86	0.86	0.86
mountain	1.00	0.71	0.83
plank	0.71	0.71	0.71
seatedforwardbend	0.42	0.71	0.53
tree	1.00	0.71	0.83
trianglepose	0.83	0.71	0.77
warrior1	0.44	0.57	0.50
warrior2	0.67	0.57	0.62

Fig 5: Precision, recall (average accuracies), and F1-score of the Final model on the test set.

According to the F1-scores, classes *bridge*, *childs*, *seatedforwardbend*, *warrior1*, and *warrior2* were the most difficult to identify correctly. Using **Fig 6** and **Fig 7**, further inspection are made below to find out why these classes face more difficulties.

Incorrect Labels:	True Label	Classified Label
Index 0	warrior2	seatedforwardbend
Index 1	bridge	warrior1
Index 6	childs	seatedforwardbend
Index 9	plank	seatedforwardbend
Index 11	warrior2	downwarddog
Index 12	tree	warrior2
Index 15	bridge	plank
Index 16	seatedforwardbend	childs
Index 18	trianglepose	warrior1
Index 19	childs	seatedforwardbend
Index 20	downwarddog	seatedforwardbend
Index 21	warrior1	warrior2
Index 26	warrior1	trianglepose
Index 29	trianglepose	seatedforwardbend
Index 35	seatedforwardbend	childs
Index 39	warrior1	bridge
Index 40	mountain	bridge
Index 42	warrior2	plank
Index 44	tree	warrior1
Index 51	plank	warrior1
Index 54	bridge	childs
Index 64	mountain	warrior1
Index 65	childs	seatedforwardbend

Fig 6: True labels and predicted labels of the incorrect predictions.

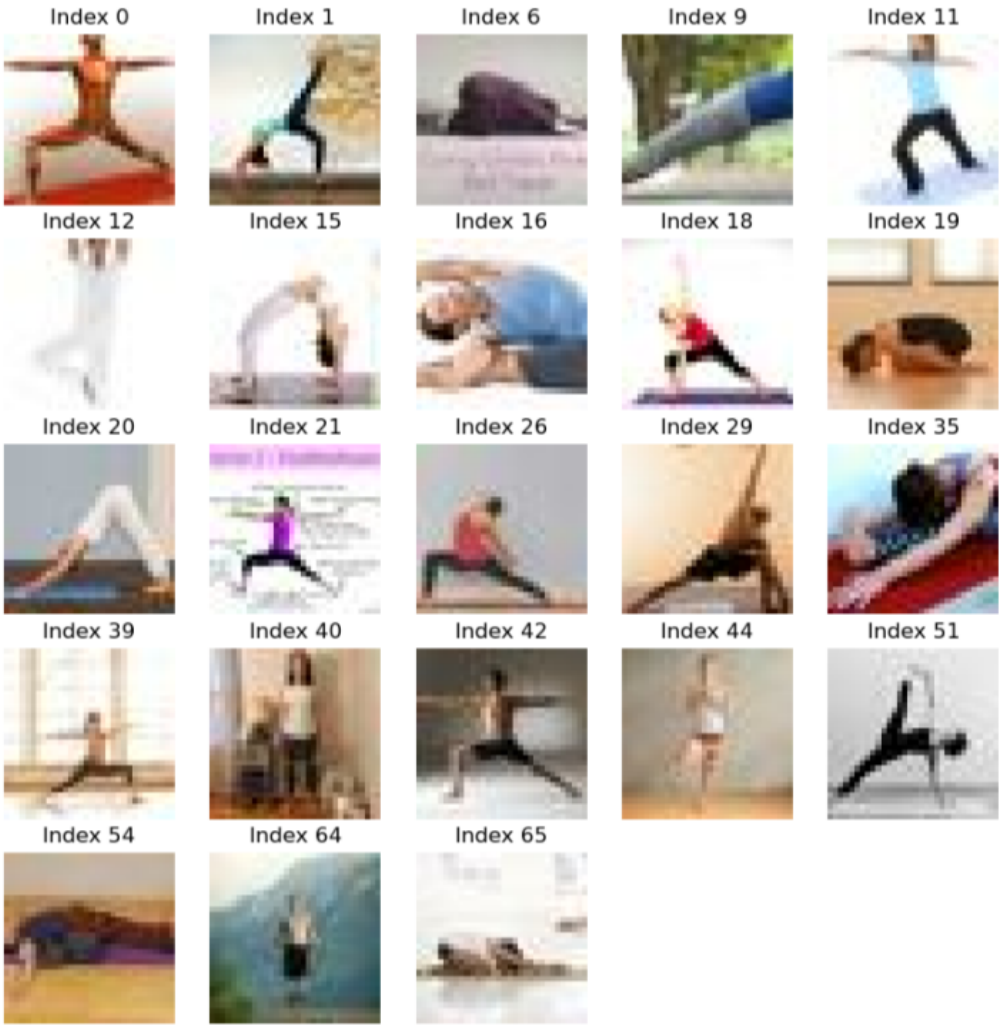

Fig 7: The images of the incorrect predictions with their index numbers.

Between *childs* and *seatedforwardbend*, majority of the errors are due to how similar the two poses are to each other, with the main difference seeming to be the leg placement. Such examples include **indexes 6, 19, and 65** where *childs* were misclassified as *seatedforwardbend* and **indexes 16 and 35** where *seatforwardbend* were misclassified as *childs*. The model might have focused more on learning that the two poses usually face downwards and not care about the legs as much.

It seems that these difficult to classify poses often have different variations where some are very similar to one or more of the other poses. Examples include **index 1** (a *bridge* with one leg up that looks like a rotated *warrior1*), **index 26** (a *warrior1* that looks like a *trianglepose* without the arm up), **index 51** (a complicated side *plank* that looks like a rotated *warrior1*), and **index 54** (a *bridge* that looks like a *childs* because of the legs). Images that include these kind of variations act as 'trick questions' to really test how well your model has learned.

According to **Fig 5**, the classes with low F1-scores tend to also be the ones with low precisions. This means that these classes are more generalisable (least distinct) and the model often has a difficult time differentiating them apart from the other poses. For e.g., *mountain* and *tree* are shown to have precisions of 1.0, indicating the model is confident in its understanding of the two poses. However, in **indexes 12, 44, and 64** it misclassifies them as *warrior2*, *warrior1*, and *warrior1* respectively. The less certain the model is about classifying a class the more sensitive its predictions are on classes that it is more certain about.

There are definitely some mistakes that are clearly due to the model not learning properly such as in **indexes {0, 11, 15, 20, 39, 40, 42, 64}** where the images look nothing like their predicted classes. Increasing the max number of epochs, gathering more data for training, and even enhancing the images' resolutions can improve upon these mistakes by deepening the learning of the model.

3. **Visualisation:**
   Ten randomly selected test images from the test set (**indexes = [56, 11, 40, 53, 66, 25, 5, 8, 55, 31]**) were put through a k-nearest neighbour (KNN) algorithm (k = 5) to find each of their five 'nearest' neighbours in the final model's learned feature space in the Flatten layer, where each image has 600 features. The distance measurement used is the Euclidean distance between each of the test images and the training images. **Fig 8** shows the results, where it starts with test image 56 in the first row, test image 11 in the second, and so on. In each row, the left most image is the test image while the other five are its 'nearest' neighbours.

   Some of the discussions made above are observed to be present in the results. It is supported in **row 1 and 10** that the model indeed has trouble differentiating *childs* and *seatedforwardbend* poses, focusing more on the body facing downwards and not the legs. What further proves this is that both rows' five nearest neighbours are exclusively have *childs* and *seatedforwardbend.*

   Through the **rows 4 and 6**, it becomes more clear that the model also has difficulty in differentiating *warrior1* and *warrior2*, where instead the model may be focusing too much on the leg positions and disregarding the hand positions. The rows's nearest neighbours are also exclusively those two classes.

   It is shown in **rows 8 and 9** that the *tree* and *mountain* test images both only have their own kind as their five nearest neighbours. This verifies their precisions of 1.0 from **Fig 5** and the model's certainty when it comes to classifying these two classes; no false positives. **Row 7** follows closely with 4/5 of the *downwarddog* test image's neighbours also being *downwarddog*, seemingly inline with its 0.86 precision (even so, *trianglepose* looks similar).

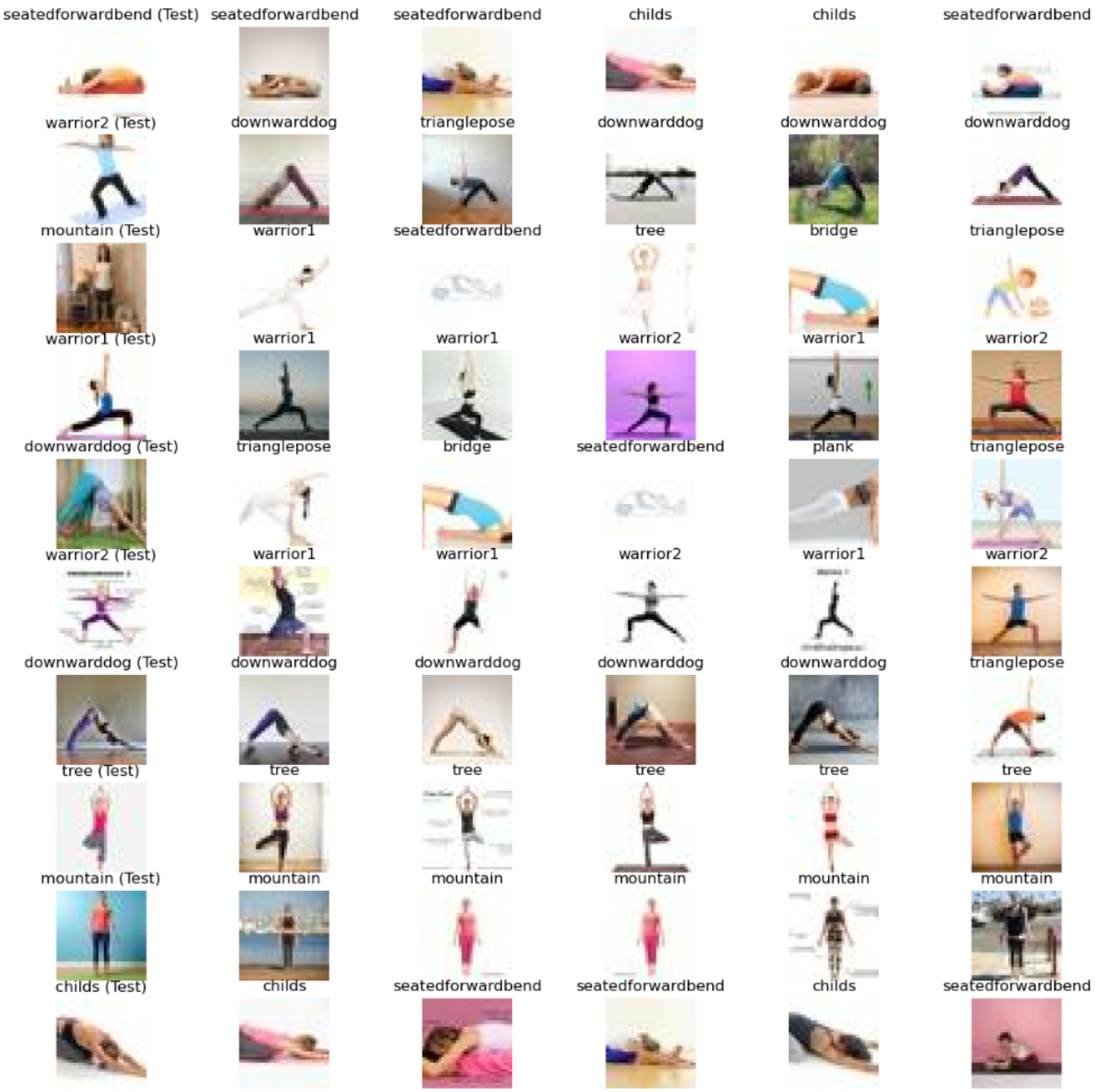

Fig 8: KNN (k = 5) results of ten randomly selected test images (indexes mentioned above).

It is also learnt that some classification errors are due to noise in the images. In **rows 3 and 5**, to the human eye it is fairly clear what poses are depicted. However, none of the nearest neighbours of the two rows match the respective test image class. This is likely because of the distracting and non-simple backgrounds illustrated in the images that the model may have identified as part of the supposed pose. It cannot be a coincidence that the other 8 rows have clearer backgrounds and more neighbours that match the test image class.

With **row 2** being the only result with no obvious explanation, the model seems to have learned a suitable enough feature space for majority of the classes and even though it may not be able to tell apart some class pairs it at least groups them together and knows they are similar. In the case of the noise errors, it indirectly yet quite clearly informs humans of why those images are difficult to classify and provides the opportunity to fix it.