Logo

Description automatically generated with medium confidence

**Module Title:** Human-Robot Interaction

**Module Code:** EE4705

**Project No.:** 1

**Title of Lab Exercise:** Project 1 Gesture Recognition

**Module Group No.:** EE4705-A

**Student Contributed to This Assignment:**

| Admission Number | Name |
| --- | --- |
| A220837H | Zheng Jiezhi |
| A0184893L | Ng Wei Jie, Brandon |
| A0248558M | Zhao Hengyuan |

**Date of Lab Exercise:** 5th February 2022

**Module Lecturer:** Professor Shuzhi Sam Ge

| **Task** | **Item** | **Score distribution** | **Name** |
| --- | --- | --- | --- |
| **Task 1** | **1** | **20%** | **Zhao Hengyuan** |
|  | **2** | **20%** | **All** |
|  | **3** | **20%** |  |
| **Task 2** | **1** | **20%** | **Zheng Jiezhi** |
|  | **2** | **20%** | **All** |
|  | **3** | **20%** |  |
| **Task 3** | **1** | **20%** | **Brandon** |
|  | **2** | **20%** | **All** |
|  | **3** | **40%** |  |

**Task 1**

**1.1. Literature Review**

**1.1.1 Gesture Recognition**

In computer science, gesture recognition is an area of study involving identification of human gestures based on mathematical algorithms. In a broader sense, the training data for gesture recognition usually comes from the movements of different parts of the human body. More particularly, it refers to movements of the face and hands. Users can use simple gestures to control or interact with devices, allowing computers to understand human behavior. Its core technologies are gesture segmentation, gesture analysis and gesture recognition [1].

**1.1.2 The Applications of Gesture Recognition in Human-Robot Interaction (HRI)**

The current focus in the field includes emotion recognition from face and gesture recognition. Users can use simple gestures to control or interact with devices without touching them and thus it has wide applications in Human-Robot Interaction. The recognition of posture, gait and human behavior is also the focal point of gesture recognition technology. Gesture recognition can be seen as a way for computers to understand human language, thereby building a richer bridge between machines and humans than raw textual user interfaces or even GUIs (Graphical User Interfaces).

Gesture recognition enables people to communicate with machines (HMIs) and interact naturally without any mechanical (physical) assistant devices. Using the concept of gesture recognition, a finger can be detected and synchronically shown at the computer screen so that the cursor moves accordingly [2]. This can make conventional input devices such as mice, keyboards and even touch screens redundant.

**Gesture input and recognition:** Use a high-definition camera as the acquisition method to ensure the detection and recognition of (unit depth of field) within the range of vision and range. The detection process adopts a non-contact method, and the target and recognition rate are high. The algorithm analyzes and processes the input image to detect the target (type).

**Noise Removal and Information Enhancement:** From video stream input, to gesture recognition, to information conversion, each link of these processes may be affected by equipment electromagnetic interference, algorithm limitations and other aspects. These interferences and influences will eventually form data noise, which will affect the noise. Improper handling will distort the gesture image, thereby affecting the final interactive instruction.

**Gesture Segmentation and Feature Extraction:** The gesture model database is established through the implementation, and then the gestures in the data stream are segmented and feature extracted according to the database. This paper mainly adopts serial boundary segmentation technology and parallel region segmentation technology to realize the feature parameter extraction of the gesture model, and finally forms the final interactive command according to the feature registration of the gesture model database. In the gesture recognition process, gesture feature vectors can be established according to data such as image edge pixel values, gesture outlines, gesture shapes, motion directions, and bones. Firstly, the background of the image is separated from the gesture image by the background separation algorithm, and then each dimension attribute in the gesture image is stripped from the data stream and the gesture image in the feature library is mapped to the feature vector to complete the feature extraction of the gesture image.

**Feature classification:** After the gesture features are extracted, the gestures are classified by the pattern recognition intelligent algorithm, and finally correspond to a set of specific function commands. In this paper, a decision tree based on statistics is used as the classifier. The decision tree actually uses the idea of ​​"divide and conquer", combined with the established gesture-command comparison table, and uses the decision tree algorithm to traverse the data table to compare the gesture features and functions. Commands are matched, and the conversion from gestures to antacid commands is finally completed.

**1.2. Neural Network Architecture**

The objective of this task is to classify hand gestures using fully-connected layers. The dataset contains 4 classes - left, right, peace and palm. The class distribution of the dataset is shown in Figure 1.2.1. The dataset is imbalanced where two classes have about 25 images and the other two classes have about 15 images.

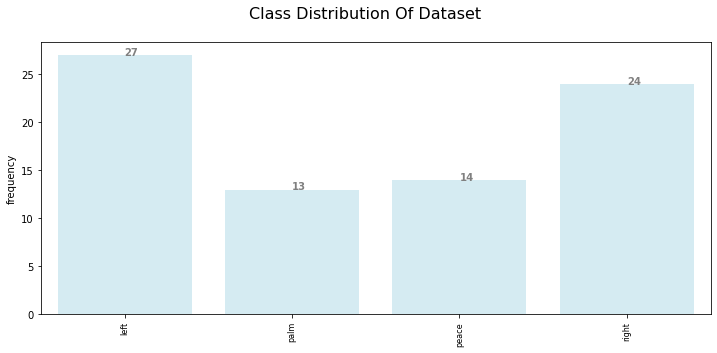
****

Figure 1.2.1 shows the class distribution of the dataset

The image is resized to height and width of 48 by 48 pixels and converted from RGB to YCbCr. A mask is used to obtain hand pixels if the pixel value falls within a threshold. Figure 1.2.2 shows how the image is preprocessed to train the model.

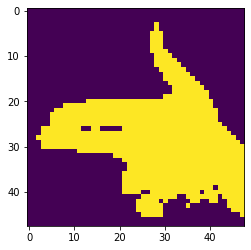


Figure 1.2.2 shows the preprocessed image

4 models are explored for hand gesture classification. The first model has 1 fully-connected block. The Linear layer in the fully-connected block is followed by a BatchNorm1d and a ReLU. The first model is designed to keep the network small to fit on the dataset. As the classification task does not seem complicated, a first model is designed as a lightweight baseline. The second model contains 3 fully-connected blocks. Given more layers, the network is still relatively small but it serves to compare the model’s ability to generalize better than the baseline. The third model has the same number of fully-connected blocks as the first model; however, the number of nodes in the full-connected block is more to understand how the model extracts features better given more nodes. The fourth model is an adaptation of the first model by adding a dropout regularizer to prevent overfitting. Note that all the model ends with a Linear layer with the number of output nodes equal to the number of classes and a Softmax for fair comparison of the performance. The model is trained on Google Colab.

| Model 1   | Linear(2304,32)  BatchNorm1d(32)  ReLU() | | --- | | Linear(32,4)  Softmax() | | Model 2   | Linear(2304,32)  BatchNorm1d(32)  ReLU() | | --- | | Linear(32,16)  BatchNorm1d(16)  ReLU() | | Linear(16,8)  BatchNorm1d(8)  ReLU() | | Linear(8,4)  Softmax() | | Model 3   | Linear(2304,512)  BatchNorm1d(512)  ReLU() | | --- | | Linear(512,4)  Softmax() | | Model 4   | Linear(2304,512)  BatchNorm1d(512)  Dropout(0.5)  ReLU() | | --- | | Linear(512,4)  Softmax() | |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |

Table 1.2.1 shows how the models explored for hand gesture classification

**1.3. Model Performance**

Model 1 has an accuracy of 0.8750. Model 2 has an accuracy of 0.9167. Model 3 has an accuracy of 0.9167. Model 4 has an accuracy of 0.9167. Model 2 performs better than Model 1 because it has more layers to generalize the data better. Model 3 performs better than Model 2 because it has more nodes for better feature extraction. Model 4 performs better than Model 1 but no better than Model 2 and Model 3 with the regularizer to prevent overfitting. Model 1 serves as a baseline and performs average given that the deep learning task is simple. Despite modifications to the baseline models by adding more layers, nodes, and regularizer, the model performance is stuck at 0.9167. This might be because the model is not able to learn the spatial information in the image when the pixels of the image are reshaped from 2D to 1D array. The confusion matrix of the 4 models is shown in Figure 1.3.1. The models generally have trouble differentiating left and right likely due to the loss in spatial information to differentiate the images. Refer to Jupyter Notebook for the training and validation losses.

| Model 1 Confusion Matrix | Model 2 Confusion Matrix |
| --- | --- |
| Model 3 Confusion Matrix | Model 4 Confusion Matrix |

Figure 1.3.1 shows how the accuracy and normalized confusion matrix for the 4 models

**Task 2**

**2.1 Convolutional Neural Network**

Convolutional Neural Networks (CNN) is a class of Feedforward Neural Networks (Feedforward Neural Networks) that includes convolutional computation and has a deep structure, and is one of the representative algorithms of deep learning [3][4] . Convolutional neural networks have the ability of representation learning and can perform shift-invariant classification of input information according to its hierarchical structure, so it is also called "shift-invariant artificial neural network". Neural Networks, SIANN)” [5].

The research on convolutional neural networks began in the 1980s and 1990s, and the time delay network and LeNet-5 were the earliest convolutional neural networks [6]; after the 21st century, with the introduction of deep learning theory And the improvement of numerical computing equipment, the convolutional neural network has been developed rapidly, and has been applied to the fields of computer vision, natural language processing and so on [5].

Convolutional neural networks are constructed by imitating the visual perception mechanism of biology, and can perform supervised learning and unsupervised learning. Small computational effort to learn grid-like topology features such as pixels and audio, with stable effects and no additional feature engineering requirements on the data [3][4].

**Input Layer:** The input layer of a convolutional neural network can handle multi-dimensional data. Commonly, the input layer of a one-dimensional convolutional neural network receives one-dimensional or two-dimensional arrays, where one-dimensional arrays are usually time or spectral samples; two-dimensional arrays may contain multiple channel; the input layer of the two-dimensional convolutional neural network receives two-dimensional or three-dimensional arrays; the input layer of the three-dimensional convolutional neural network receives four-dimensional arrays [7]. Since convolutional neural networks are widely used in the field of computer vision, many studies presuppose three-dimensional input data when introducing their structure, that is, two-dimensional pixels and RGB channels on a plane.

**Hidden Layer:** The hidden layer of the convolutional neural network includes three common structures: convolutional layer, pooling layer and fully connected layer. In some more modern algorithms, there may be complex structures such as the Inception module and residual block. In common constructions, convolutional and pooling layers are specific to convolutional neural networks. Convolution kernels in convolutional layers contain weight coefficients, while pooling layers do not, so pooling layers may not be considered separate layers in the literature. Taking LeNet-5 as an example, the order in which the three common types are constructed in the hidden layer is usually: input - convolution layer - pooling layer - fully connected layer - output.

Convolutional layer parameters: The parameters of the convolutional layer include the size of the convolution kernel, the stride and the padding, which together determine the size of the output feature map of the convolutional layer, which is the hyperparameter of the convolutional neural network [3]. The size of the convolution kernel can be specified as an arbitrary value smaller than the size of the input image. The larger the convolution kernel, the more complex the input features that can be extracted [3] .

**Fully-Connected Layer:** The fully connected layer in a convolutional neural network is equivalent to the hidden layer in a traditional feedforward neural network. The fully connected layer is located in the last part of the hidden layer of the convolutional neural network and only transmits signals to other fully connected layers. The feature map loses its spatial topology in the fully connected layer, is expanded into a vector and passes through the activation function [3] .

From the point of view of representation learning, the convolutional layer and pooling layer in the convolutional neural network can perform feature extraction on the input data, and the function of the fully connected layer is to nonlinearly combine the extracted features to obtain the output, that is, the fully connected layer itself It is not expected to have feature extraction capabilities, but attempts to utilize existing high-order features to accomplish the learning objective.

**Output Layer:** The upstream of the output layer in a convolutional neural network is usually a fully connected layer, so its structure and working principle are the same as the output layer in a traditional feedforward neural network. For image classification problems, the output layer uses a logistic function or a normalized exponential function (softmax function) to output the classification labels [16]. In the object detection problem, the output layer can be designed to output the center coordinates, size and classification of objects [7]. In image semantic segmentation, The output layer directly outputs the classification result of each pixel [7].

**2.2 Neural Network Architecture**

The objective of this task is to classify hand gestures using convolutional layers. The same dataset is used. Different preprocessing methods are applied on the image to provide richer features to train the model. The first preprocessing method resizes to height and width of 48 by 48 pixels and converts from RGB to YCbCr to extract the hand pixels in the image. The second preprocessing method converts from RGB to grayscale and resizes to height and width of 48 by 48 pixels. Grayscale provides richer information about the edges of the hand pixels. The third preprocessing method only resizes the image to height and width of 48 by 48 pixels and left to the model to extract the features.

3 models are explored for hand gesture classification. The first model has 2 convolutional blocks. Each block has 1 Conv2d, 1 BatchNorm2d, 1 ReLU, and 1 MaxPool2d. The first model is designed to keep the network small and serves as a baseline model. For the first model, the input is the hand mask image by applying the first preprocessing method described in Task 2.2. The first convolutional model can also be compared against the first fully-connected model since the processing steps are the same. The second model uses the same architecture as the first model except that the input is a grayscale image by applying the second preprocessing method. The model is provided richer information from a grayscale image than a hand mask image. The third model has 3 convolution layers and serves to compare the model’s ability to generalize better than the second model. The input is a grayscale image. The fourth model has the same number blocks as the first model except that the input layer takes in 3 channels instead of 1 channel to take in a RGB image as input. Note that all the model ends with a Flatten, a Linear layer and Softmax for fair comparison of the performance. The model is trained on Google Colab.

| Model 1  (Hand Mask Input )   | Conv2d(1,4,3)  BatchNorm2d(4)  ReLU()  MaxPool2d(2,2) | | --- | | Conv2d(4,8,3)  BatchNorm2d(8)  ReLU()  MaxPool2d(2,2) | | Flatten() | | Linear(800,4)  Softmax() | | Model 2  (Grayscale Image Input)   | Conv2d(1,4,3)  BatchNorm2d(4)  ReLU()  MaxPool2d(2,2) | | --- | | Conv2d(4,8,3)  BatchNorm2d(8)  ReLU()  MaxPool2d(2,2) | | Flatten() | | Linear(800,4)  Softmax() | | Model 3  (Grayscale Image Input)   | Conv2d(1,4,3)  BatchNorm2d(4)  ReLU()  MaxPool2d(2,2) | | --- | | Conv2d(4,8,3)  BatchNorm2d(8)  ReLU()  MaxPool2d(2,2) | | Conv2d(8,8,3)  BatchNorm2d(8)  ReLU()  MaxPool2d(2,2) | | Flatten() | | Linear(128,4)  Softmax() | | Model 4  (RGB Image Input)   | Conv2d(3,4,3)  BatchNorm2d(4)  ReLU()  MaxPool2d(2,2) | | --- | | Conv2d(4,8,3)  BatchNorm2d(8)  ReLU()  MaxPool2d(2,2) | | Flatten() | | Linear(800,4)  Softmax() | |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |

Table 2.2.1 shows how the models explored for hand gesture classification

**1.3. Model Performance**

Model 1 has an accuracy of 0.8333. Model 2 has an accuracy of 0.9583. Model 3 has an accuracy of 1.0000. Model 4 has an accuracy of 1.0000. The convolution baseline Model 1 did not perform better than the fully-connected baseline Model 1. The reason could be due to the number of convolutional layers not being enough to extract useful information from the image. However, a compact model can perform well given good data. Model 2 has the same architecture as Model 1 but was trained on grayscale images as inputs. Model 2 performs at least 0.1 better than Model 1, beating all the fully-connected models in terms of performance. This could be due to richer information in the grayscale image than the hand mask image. The performance can be improved further by adding more layers. Model 3 achieved 1.0000 on the test dataset but Model 2 did not. This is unlike the performance of the fully-connected models that is stuck at 0.9167. Model 4 performs as well as Model 3 with fewer layers but was trained on RGB images instead. The experiments in this task show how convolution models easily beat fully-connected layers in performance. A small, compact model can perform equally well given good data to learn.

| Model 1 Confusion Matrix | Model 2 Confusion Matrix |
| --- | --- |
| Model 3 Confusion Matrix | Model 4 Confusion Matrix |

Figure 2.3.1 shows how the accuracy and normalized confusion matrix for the 4 models

**Task 3**

**3.1. Literature Review**

**3.1.1. Dataset Collection**

**HGM-4 Mendeley Dataset:** There are 4160 images of 26 hand gestures in the dataset. The images were captured indoors by 4 cameras using the same laptop camera but at different positions. Images taken using the one camera were used for testing while the rest were used for training. The background of the hand-gesture images was removed. The benefit of this dataset is that it provides multi-view images of gestures as there are few datasets dealing with this problem. Since the gestures in this dataset represent the alphabet letter of Vietnamese sign language, the dataset can be applied on contactless device control and sign language interpretation applications. [8]

**DSV128 Gesture Dataset:** There are 1342 images of 11 hand and arm gestures in the dataset. The data was collected from 29 subjects under 3 different light conditions for a total of 122 trails. The subject stood in front of a stationary background and performed the action for 6 seconds. Data from 23 subjects were used for training and 6 subjects were for testing. The benefit of this dataset is that it uses a DVS128 camera instead of a RGB camera that generates images when there is a change in magnitude in that pixel; hence, it operates at a higher frame rate and is more energy efficient than RGB camera. The dataset can be applied on low power IoT devices to classify hand and arm gestures. [9]

**EgoGesture Dataset:** There 2953224 frames of 83 hand gestures in the dataset. The data was collected using Intel RealSense SR300 from 50 subjects who performed the action in 6 scenes - 4 indoors and 2 outdoors. The scenes included a dynamic background in which the subject moves. The realsense is strapped on the subject’s head to capture first-person images. The subjects were split to ratio 3:1:1 for training, validation, and testing. The benefit of this dataset is that it provides depth maps as features that help to classify gestures that are challenging from RGB images alone. Another benefit is that the dataset provides raw videos that allow gesture classification from temporal information. As the dataset provides first-person images, this dataset can be applied on VR/AR and contactless device control applications. [10]

**3.1.2. Deep Learning Models**

**Multichannel Convolutional Neural Network (MCNN):** MCNN has two convolutional layers, each followed by a max pooling layer. The first convolutional layer uses a cubic kernel instead of 2D kernel to improve feature extraction and has in 3 channels for grayscale raw image, Sobel horizontal-filtered image, and Sobel vertical-filtered image. The Sobel filter provides edge enhancement images for feature extraction. After the second max pooling layer, the nodes are flatten and passed to two fully-connected layers to classify the hand posture. The benefit of this method is that the model is very small and is trained on preprocessed images. The application of this method is on gesture classification problems that require real-time inference. [11]

**HGR-Net:** HGR-Net has two stages to output a hand segmentation map and the gesture classification. The first stage uses an ASPP module to encode features at multiple scales and decoded using convolutional layers to obtain the segmentation maps. The second stage uses two 4-layers convolutional blocks where the first block receives the RGB image and the second block receives the segmentation image from ASPP to learn the appearance and shape features respectively. The features extracted from these two blocks in the second stage are flattened, passed through two fully-connected layers and fused to classify the gestures. The benefit of this method is that it can work in different environments as the model will segment the hand and use the segmentation map as features to classify the gesture. The application of this method can be on many other gestures dataset that might be better off segmenting the hand pixels for useful feature extraction. [12]

**ResNet50:** ResNet is one of the state-of-art models for image classification tasks. ResNet solves the vanishing gradient problem through skip connections in the residual blocks that many deep learning models fail to when the error is very small. [14] The number in the model name indicates the number of layers. For example, there are 50 layers in ResNet50. ResNet is trained on a large dataset and is used for transfer learning by removing the last full-connected layer. ResNet50 serves as a backbone for feature extraction to classify gestures. [13] The benefit of using ResNet is for transfer learning to train a new classifier with less data and fewer epochs. ResNet can be applied in any image classification tasks to create a new classifier without training from scratch.

**3.2. Data Collection**

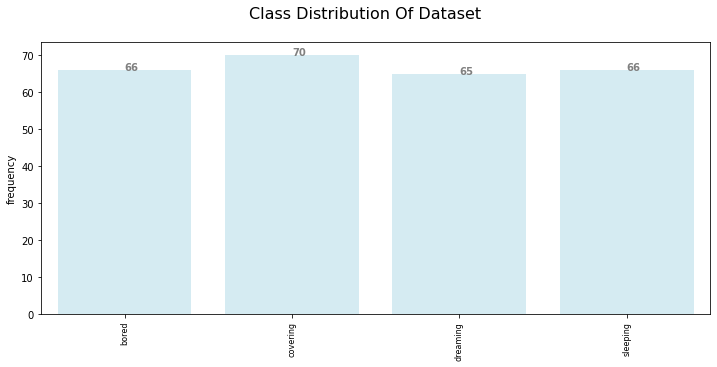


Figure 3.2.1 shows the class distribution of the dataset

The objective of this task is to classify body postures and behavior. The dataset contains 4 classes - bored, covering, dreaming, and sleeping. The class distribution of the dataset is shown in Figure 2.1. The dataset is balanced, and each class contains about 65 data points.

Figure 3.2.2 shows a sample image of each class in the dataset. The images are taken at 13 different locations. At each location, 5 images at different angles are taken for each action. The purpose of taking images at different angles and locations is to help the model generalize and learn the body posture and behavior from the images.

| Bored | Covering | Dreaming | Sleeping |
| --- | --- | --- | --- |

Figure 3.2.2 shows a sample image of each class in the dataset

The images are taken using a mobile phone camera using aspect ratio 1:1. The images are resized to dimension 360 by 360 pixels and stored in the corresponding sequence and action directory. Sequence refers to the location the image is taken at. Action refers to the class. For example, the image seq1/bored/IMG\_20220211\_201536.jpg is taken at “Location 1”, capturing the person performing the action “bored”.

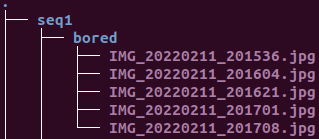


Figure 3.2.3 shows how the image are stored in the dataset

**3.3. Neural Network Architecture**

The important training details and description are highlighted in Table 3.3.1. The model is trained on Google Colab.

| Training Detail | Description |
| --- | --- |
| Input Size | Input RGB image is resized to 256 by 256 pixels and fed to the model. |
| Optimizer | Adam optimizer converges well and does not require much hyperparameter tuning. |
| Loss Function | CrosEntropyLoss is typically used for classification problems. |
| Learning Rate | Learning rate is set to 0.0001 to allow the model to converge fast without fluctuating rapidly when the model converges. |
| Data Split | Data from sequence 1 to 10 are used for training while data from sequence 11 to 13 are used for testing. This is to prevent overlap in data for training and testing. |
| Augmentation | RandomVerticalFlip, RandomHorizontalFlip, ColorJitter, RandomRotation are applied to augment the image to help the model generalize better since the dataset is relatively small. |

Table 3.3.1 shows how the training details for the models

4 models that are explored for body posture and behavior classification. The first model has 3 convolutional blocks. Each block has 1 Conv2d, 1 BatchNorm2d, 1 ReLU, and 1 MaxPool2d. The first model serves as a baseline and is designed to keep the network small and fit the model on the dataset. The second model contains 5 convolutional blocks instead of 3. The network is still relatively small but it serves to compare the model's ability to generalize against the first model. The third model has the same number of convolutional blocks as the second model; however, the number of channels to improve the feature extraction. The fourth model is to compare the custom models against state-of-art models, in particular ResNet18. The objective is to understand how transfer learning can reduce the time needed for the model to learn on new datasets. However, ResNet18 has 18 layers, 3 times more layers than the custom models. Note that all the model ends with a Flatten, a Linear layer and Softmax for fair comparison of the performance.

| Model 1   | Conv2d(3,8,3)  BatchNorm2d(8)  ReLU()  MaxPool2d(2,2) | | --- | | Conv2d(8,16,3)  BatchNorm2d(16)  ReLU()  MaxPool2d(2,2) | | Conv2d(16,32,3)  BatchNorm2d(32)  ReLU()  MaxPool2d(2,2) | | Flatten() | | Linear(28800,4)  Softmax() | | Model 2   | Conv2d(3,8,3)  BatchNorm2d(8)  ReLU()  MaxPool2d(2,2) | | --- | | Conv2d(8,16,3)  BatchNorm2d(16)  ReLU()  MaxPool2d(2,2) | | Conv2d(16,32,3)  BatchNorm2d(32)  ReLU()  MaxPool2d(2,2) | | Conv2d(32,64,3)  BatchNorm2d(64)  ReLU()  MaxPool2d(2,2) | | Conv2d(64,128,3)  BatchNorm2d(128)  ReLU()  MaxPool2d(2,2) | | Flatten() | | Linear(4608,4)  Softmax() | | Model 3   | Conv2d(3,16,3)  BatchNorm2d(16)  ReLU()  MaxPool2d(2,2) | | --- | | Conv2d(16,32,3)  BatchNorm2d(32)  ReLU()  MaxPool2d(2,2) | | Conv2d(32,64,3)  BatchNorm2d(64)  ReLU()  MaxPool2d(2,2) | | Conv2d(64,128,3)  BatchNorm2d(128)  ReLU()  MaxPool2d(2,2) | | Conv2d(128,256,3)  BatchNorm2d(256)  ReLU()  MaxPool2d(2,2) | | Flatten() | | Linear(9216,4)  Softmax() | | Model 4   | ResNet18()  pretrained=True | | --- | | Flatten() | | Linear(512,4)  Softmax() | |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |

Table 3.3.2 shows how the models explored for body posture and behavior classification

Model 1 has an accuracy of 0.9344. Model 2 has an accuracy of 0.9672. Model 3 has an accuracy of 98.36. Model 4 has an accuracy of 1.0000. Model 2 performs better than Model 1 because it has more layers to generalize the data better. Model 3 performs better than Model 2 because it has more channels for better feature extraction. Model 1 is the baseline model and it achieved good accuracy. Having more layers and channels allows better feature extraction and higher accuracy but at the cost of a heavier model or an overfitted model given that the dataset is small. Model 4 performs the best and achieves 1.0000 accuracy within 10 epochs. This demonstrates how transfer learning helps models to learn new datasets in a few epochs. The normalized confusion matrices for the four models are shown in Figure 3.3.1. From the confusion matrix, the models achieve high accuracy for the 4 classes. Classifying the images between bored and covering might be challenging to the models since the images for covering might be taken at the side. The model might have only seen the one hand on the face instead of two for the covering images similar to the one hand on the face for the bored images. Refer to Jupyter Notebook for the training and validation losses.

**3.4. Problems Encountered**

The first problem encountered was model overfitting. Overfitting occurred for the custom models and ResNet18(pretrained=False) but did not occur for ResNet18(pretrained=True). The model did better under transfer learning. The model might not be robust against background noises since the dataset is much smaller than the dataset that the pretrained model was trained on. Besides, the images were only collected from 4 locations, and the person in the image is small. The solution was to increase the variability of the dataset. The images were collected from 13 different locations and were captured at different angles. The image is also zoomed and focused on the person.

The second problem I encountered was that the training accuracy increases much faster than the testing accuracy. Ideally, the training accuracy should increase closely with the testing accuracy. Since the dataset is small, the model learns from the same but small training data each epoch. Hence, the model learns the training data fast but fails to generalize on the testing data, resulting in the difference in increase in accuracy. The solution was to train the model on an augmented dataset. Online augmentation was implemented to synthetically increase the dataset by adjusting the images such as brightness, contrast, and mirroring. However, in Task 1 and 2, mirroring and rotation augmentations cannot be used as the left and right hands are symmetric; the model will fail to learn the data.

| Model 1 Confusion Matrix | Model 2 Confusion Matrix |
| --- | --- |
| Model 3 Confusion Matrix | Model 4 Confusion Matrix |

Figure 3.3.1 shows how the accuracy and normalized confusion matrix for the 4 models

**References**

**Task 1**

[1]Pavlovic, V., Sharma, R. & Huang, T. (1997), "Visual interpretation of hand gestures for human-computer interaction: A review", IEEE Transactions on Pattern Analysis and Machine Intelligence, July, 1997. Vol. 19(7), pp. 677 -695.

[2] Chen, Shijie; "Gesture Recognition Techniques in Handwriting Recognition Application", Frontiers in Handwriting Recognition p 142-147 November 2010

**Task 2**

[3] Goodfellow, I., Bengio, Y., Courville, A.．Deep learning (Vol. 1)．Cambridge：MIT press，2016：326-366

[4] Gu, J., Wang, Z., Kuen, J., Ma, L., Shahroudy, A., Shuai, B., Liu, T., Wang, X., Wang, L., Wang, G. and Cai, J., 2015. Recent advances in convolutional neural networks. arXiv preprint arXiv:1512.07108.

[5] Zhang, W., 1988. Shift-invariant pattern recognition neural network and its optical architecture. Proceedings of the annual conference of the Japan Society of Applied Physics.

[6] LeCun, Y. and Bengio, Y., 1995. Convolutional networks for images, speech, and time series. The handbook of brain theory and neural networks, 3361(10), 1995.

[7] Ng, A., Kian, K. and Younes, B. Convolutional Neural Networks, Deep learning.

**Task 3**

[8] Hoang, Vinh. (2020). HGM-4: A new multi-cameras dataset for hand gesture recognition. Data in Brief. 30. 105676. 10.1016/j.dib.2020.105676.

[9] A. Amir et al., "A Low Power, Fully Event-Based Gesture Recognition System," 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017, pp. 7388-7397, doi: 10.1109/CVPR.2017.781.

[10] Y. Zhang, C. Cao, J. Cheng and H. Lu, "EgoGesture: A New Dataset and Benchmark for Egocentric Hand Gesture Recognition," IEEE Transactions on Multimedia (T-MM), Vol. 20, No. 5, pp. 1038-1050, 2018.

[11] Barros, Pablo & Magg, Sven & Weber, Cornelius & Wermter, Stefan. (2014). A Multichannel Convolutional Neural Network for Hand Posture Recognition. 403-410. 10.1007/978-3-319-11179-7\_51.

[12] Dadashzadeh, Amirhossein & Targhi, Alireza & Tahmasbi, Maryam. (2018). HGR-Net: A Fusion Network for Hand Gesture Segmentation and Recognition. 10.1049/iet-cvi.2018.5796.

[13] Rashmi Bakshi, "Hand Hygiene Video Classification Based on Deep Learning," 2021 CoRR

[14] K. He, X. Zhang, S. Ren and J. Sun, "Deep Residual Learning for Image Recognition," 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 770-778, doi: 10.1109/CVPR.2016.90.