CS230

CADDYIAN - Diver Gesture Language Classification

Veronica Peng
Dept. of Computer Science
tpeng24@stanford.edu

Xi Yu
Dept. of Bioengineering
isruyuki@stanford.edu

Wenxi Zhao
Dept. of Civil & Env. Engineering
wenxi99z@stanford.edu

1 Abstract

Autonomous robot companion operated through gesture-based language, CADDYIAN, helps to protect divers in dangerous underwater environments. In this paper, we explore the capability of resnet to classify CADDYIAN gestures using the public dataset from the CADDY project. For better performace, we experimented training with both original and manually balanced datasets, with 3 resolution levels 240×180, 320×240, and 480×360, and with categorical cross entropy or hinge loss. Our best single model is resnet-18 trained on dataset2500 with 320×240 image resolution and categorical cross entropy, achieving a 97.85% test accuracy. Our best ensemble model is majority voting, achieving 98.12% test accuracy. tSNE analysis indicates that true negative class (with no gestures) can be easily confused with other classes. Additional error analysis identifies 6 major error factors, and proposes generating more training data and prior hand localization to further boost performance.

2 Introduction

To combat the risks underwater, autonomous robots accompany the divers when they operate in harsh and poorly monitored environments. To help robots and divers to communicate, a unique gesture-based communication language, CADDYIAN, was developed and tested during a pioneering underwater project funded by the European community[1]. The goal of this project is to test out a deep learning model that takes in a gesture image and outputs a prediction of gesture posed by the diver.

3 Related work

Gesture recognition has become the center of human-robot interaction and draws a lot of attention from scholars. Some gesture recognition models take video clips as input and use dense trajectory extraction and motion boundary descriptors to capture the motion and then feed the data into classifiers [2][3]. Some models take in depth images of the whole body and sort each pixel to different body parts with a randomized decision forest classifier followed by articulated feature extraction with convolutional neural networks (CNN) [4][5]. A recent paper proposed a multi-fusion model that incorporated all the techniques above, taking depth video, intensity video, and mocap data all as inputs through three separate modalities, and finally, fused them into one shared hidden layer. Randomly dropping separate channels from different modalities gave the model extra robustness [6]. Each proposed model has its strength, however, the multi-model should be superior for general cases because it combines superiority of all the other models. For our problem, given that the input is image, we anticipate using resnet alone for comparably good result. Some complementary experiments were conducted upon resnet for better performance.

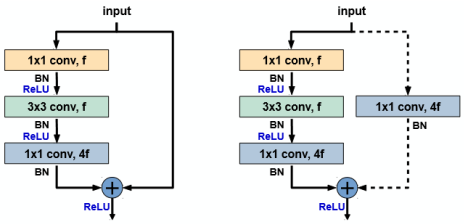
4 Dataset and Features

The CADDY Underwater Stereo-Vision Dataset contains a total of 32858 images for 16 gesture classes and one true negative class (no gesture). Classes are extremely unbalanced but well imply the frequency of gesture usage in reality. (See Figure 3 in Appendix Section 9.1) We thus randomly

CS230: Deep Learning, Winter 2018, Stanford University, CA. (LateX template borrowed from NIPS 2017.)

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
		3×3 max pool, stride 2				
conv2_x	56×56	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix} \times 2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix} \times 3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix} \times 3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix} \times 2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix} \times 4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix} \times 8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix} \times 2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix} \times 6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix} \times 6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix} \times 23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix} \times 36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix} \times 2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix} \times 3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix} \times 3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

(a) Resnet architecture



(b) Resnet shortcut.

Figure 1: Resnet architecture.

selected 4.5% (1492 images) to create the dev and test set respectively. For the training set, we tried 2 varieties to balance the data:

- **Class weight:** We use the rest 91% (29874 images) as train set. We set the *class_weight* of each sample to be proportional to the reciprocal of the number of images in its class, so that the model treats each class equally. We call this dataset **all-scenarios**.
- **Oversample + Class weight:** We randomly sample 2500 images for each class. However, 2 classes have more than 2500 images (i.e. 3308 images for *start_comm* and 13068 images with no gestures). As the model can learn many lower-level features from these data, we kept all images from these two classes. Class weight of these 2 classes is 2500 over the number of images in the class, and that of other classes is 1. We call this dataset **dataset2500**.

The output of our algorithm is a probability vector of dimension 17 for each sample. The input image size is 640×480 pixels. We performed the following processes on the input images:

- **Data augmentation:** To overcome image variability and shortage, We performed: image rotation (0° to 20 °), horizontal and vertical shifting (0 to 0.2 of the image size), zooming in and out (0.9 to 1.1 of the image size) and images normalization (with 1./255 coefficient).
- **Shrink the image:** Large images limit the batch size to 8 due to insufficient memory. To boost computation speed, we shrink the images to 240×180, 320×240 and 480×360, at which scales the gestures are still intelligible by human. With smaller images, we increase the batch size to 64.

5 Methods

5.1 The model

We used original Resnet-18, Resnet-50, Resnet-101 architecture adapted from CIFAR-10 classification task (Figure.1). In Resnet-50, one bottleneck block is composed of 1*1 conv - 3*3 conv - 1*1 conv layers. The model has an initial 7*7 conv layer, a 3*3 max pooling layer, 4 stages of repeated resnet blocks, an average pooling and a final dense softmax layer. The width and height of output halve while the number of channels roughly double with proceeding to the next stage. Within each stage, tensor size remains unchanged and the number of bottleneck blocks is variable within 3-5 range. A major enhancement of Resnet is its shortcuts that add the input to a resnet block to the output of the same block (direct summation if number of channels match, otherwise 1*1 conv to match dimensions)(Figure.1b) The shortcuts ensure unnecessary blocks easily learn the identity function. Thus, we can explore deep networks without much worries on vanishing gradients or **over-fitting**.

In terms of potential loss functions, we tried the most frequently used categorical cross entropy: $L(y, \hat{y}) = -\sum_{i=1}^{C} y \ln(\hat{y})$, and multiclass hinge loss [7]: $L(y, \hat{y}) = \sum_{j \neq y_i} max(0, s_j - s_{y_i} + \Delta)$. The hinge loss does not work as train set loss value stopped decreasing after 6 epochs.

5.2 Ensemble

In order to improve the performance of our model, we applied 2 ensemble strategies on our models:

- **Majority vote:** Each model makes a prediction on the sample. Pick the class that most model think the sample belongs to.
- **Highest precision:** Have all models calculate the train set precision for each class using **all-scenarios**. We use all-scenarios since it reflects the real distribution of classes. We then had all

models make predictions on the dev and test set. For each sample, we pick the model with the highest precision for its predicted class. For instance, in Table 1, Model 1 has the highest precision for the predicted class in the training set, so the ensemble model predicts the sample as class 4. The intuition behind the ensemble strategy is that the model is very likely to make a correct prediction on the sample when its precision is high.

Table 1: Example of the ensemble strategy highest precision

	Model 1	**Model 2**	**Model 3**
Prediction	4	16	9
Precision for the class	0.982	0.964	0.892

6 Experiments/Results/Discussion

6.1 Hyperparameter

Learning rate: We used the *keras.callbacks.ReduceLROnPlateau* to help us schedule our learning rate. We start with 0.001, and decrease the learning rate by $\sqrt{0.1}$ whenever the training set loss hasn't decrease for 5 epochs until the learning rate is smaller than 0.5×10^{-6}.

Epoch: We train our models for 50 epochs at which the model hasn't overfit train set as the dev set loss still decreases. We stopped training for two reasons: 1) dev set loss is minimal after 50 epochs 2) more time devoted to explore different models.

Batch size: We used the batch size of 64, which is the maximum batch size we can achieve without going out of memory to reduce training time on one epoch.

6.2 Metrics

Evaluation metrics include accuracy (primary metrics), weighted AUC, weighted recall, weighted precision and Tuned F1 (see Appendix 9.5).

6.3 Experiments

Experiment 1: For the first set of experiments, we used the dataset **all-scenarios**, epoch $= 50$, image size $= 240 \times 180$, batch size $= 64$. We tested three varieties of resnet for this experiment: resnet18, resnet50 and resnet101. See Table 2 below for the model performance. See Appendix Section 9 for more model performance details.

We used the resnet 18 trained on all-scenarios with image size 240×180 as a baseline because it is a simple model and small image size to start with. We expect more complicated model such as resnet-50 and larger image size such as 480×360 will improve our performance.

However, different from what we expected, resnet 18 and resnet 50 had the best test set accuracy for **all-scenarios**. Although resnet 101 is more complex, it didn't achieve a as good of a performance as simpler model. However, the worse performance is not caused by overfitting, because the dev set accuracy was still increasing at epoch 50. It is probably because resnet 101 has many more parameters so that it needs to train for longer to learn the right parameters.

Table 2: Model performance on dataset all-scenarios

	Train set accuracy	**Dev set accuracy**	**Test set accuracy**	**AUC**
resnet 18 (Baseline)	93.34%	95.10%	**95.17**%	99.37%
resnet 50	95.74%	95.37%	94.91%	99.70%
resnet 101	88.96%	89.94%	90.41%	98.75%

Experiment 2: For the second set of experiments, we used the dataset **dataset2500**, epoch $= 50$, image size $= 240 \times 180$, batch size $= 64$. We tested resnet-18 and resnet-50 for this experiment. We didn't test resnet 101 since it didn't give good performance in Experiment 1. See Table 3 below for the model performance. See Appendix Section 9.3 for more model performance details.

Both models give more accurate predictions when trained on **dataset2500** than when they are trained on dataset **all-scenarios**. However, the better performance doesn't necessarily mean *Oversample + Class weight* is a better way to balance the dataset. The dataset2500 contains more images due to

oversampling, so the better peerformance may simply be caused by having the model train on more data before it starts to overfit.

Table 3: Model performance on dataset dataset2500

	Train set accuracy	Dev set accuracy	Test set accuracy	AUC
resnet 18	96.99%	97.92%	**97.45**%	99.70%
resnet 50	96.45%	96.78%	95.84%	99.46%

Experiment 3: For the 3^{rd} set of experiments, we tested resnet 18 (proved to have the best performance by Experiment 1 and 2) on the dataset **dataset2500** with epoch $= 50$ and batch size $= 64$. We tried different input image size of 240×180 (the same resnet 18 model from Experiment 2), 320×240 and 480×360. See Table 4 for the model performance. See Appendix Section 9.4 for more model performance details. The image size 320×240 has the highest test accuracy of 97.85%, even higher than when the image size is 480×360. Higher resolution doesn't necessarily increase the performance probably because the lower resolution is good enough for both the model to differentiate the gestures so that higher resolution doesn't provide new information about the gesture.

Table 4: Resnet18 performance on dataset2500 with different input image sizes

	Train set accuracy	Dev set accuracy	Test set accuracy	AUC
240 $\times$ 180	96.99%	97.92%	97.45%	99.70%
320 $\times$ 240	96.96%	97.79%	**97.85**%	99.62%
480 $\times$ 360	96.27%	97.39%	96.98%	99.58%

Experiment 4: Ensemble We used all the single models we trained in our ensembles. According to Table 5, Highest precision strategy achieves the same performance as our best single model does, and Majority voting's performance on test set exceeds that of our best single model's by a small margin.

Table 5: Ensemble model performance

Ensemble Models	Dev set accuracy	Test set accuracy
Highest precision	97.92%	97.45%
Majority voting	97.72%	**98.12%**

6.4 Analysis on Recall and Precision

We used the resnet-18 trained on dataset2500 with image size 240×180 as an example for recall and precision analysis. The model achieved recall and precision scores above 0.95 for almost all classes. The macro averaged precision is 0.972, and the weighted averaged precision is 0.975. The macro averaged recall is 0.987, and the weighted averaged recall is 0.975. See figure 24 in Appendix Section 9.3 for further details about precision and recall for each class. In the model's confusion matrix (See Figure 20 in Appendix Section 9.3), we see most of the misclassification happens when the *trueneg* (the biggest class) are classified as other classes or when other classes are classified as trueneg. This phenomenon is a natural result of having an unbalanced dataset, and it often causes lower precision or recall for smaller classes. For example, 5 *trueneg* samples are misclassified as *boat*, which leads to a 13.2% reduction in the precision of class *boat* (a small class with 33 images), while only contributes 0.76% reduction in the recall of class *trueneg* (a large class with 656 images). In this example, we see that the model already achieves a really good performance on classifying *trueneg*, but the large number of *trueneg* images makes even a low error rate fatal.

6.5 Indication from tSNE

The tSNE plot (Figure 2) is plotted by extracting final flatten layer vector from **Resnet 50-all-scenarios**. Most of the classes take up a unique domain or a group of closely located domains with minor points misplaced on other classes' domain. One dominant mis-classification observed is that many *trueneg* (pink) points are mis-placed to other classes' domains. It is a manifestation that there is a wide range of body positions divers may take during missions as they are not communicating with robots but some of which can be easily misinterpreted as one of the CADDYIAN gestures. To improve the precision of communication, the gesture "start communication" should be set as the

trigger, and only after the robot recognizes this gesture should it be alerted and keep tracking of the diver's action. Similarly, the gesture "end communication" is equally important.

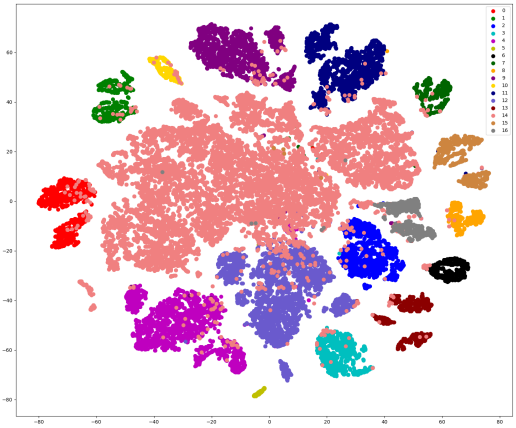


Figure 2: tSNE plot for 17 classes of gestures.

6.6 Error Analysis

An error analysis was performed on **Resnet 50-all-scenarios**. Given that mis-classification between *trueneg* and *start_com*, and *trueneg* and *end_com* should be the major concern in the real world applications, the error analysis focused specifically on those two pairs (Table 6 in Section 9.5). Among all the 166 *start_com* samples and 120 *end_com* samples in the test set, none of them are classified as *trueneg*. On the other hand, among the 656 *trueneg* samples, 13 are misclassified as *start_com*, and 6 as *end_com*, contributing 1.98% and 0.76% of reduction in the recall of *trueneg* respectively. After sorting all the misclassified samples, we found 6 major reasons that can cause the errors: 1) Similar hand gesture; 2) Dim light; 3) Similar body figure; 4) Fuzziness; 5) More than one diver; 6) Distraction. Among those 6 factors, 3 of them can be solved by implementing hand localization procedure before classification and 2 of them can be addressed by collecting more training data in the same scenarios. This analysis gives us a direction on how to improve our model.

7 Conclusion/Future Work

7.1 Conclusion

Among all of our single models, Resnet-18 trained on balanced dataset2500 with input image size of 320×240 has the best performance, achieving 97.45% test accuracy. Among our ensemble models, majority voting has the best performance, achieving 98.12% for test accuracy. Through experiments, our model can provide some insights for other underwater image classification problems: 1) image with lower resolution may serve as better input since it can weaken the effect of noise; 2) combining over-sampling and adding class-weight in the loss function can make up for the training set imbalance.

7.2 Future Work

Given the current weakness of our model, three directions we can follow to improve our model.

GAN Augmentation: CADDYIAN data collection is costly. Using GAN to generate images with the current dataset can be a possible solution to solve dataset shortage.

Nested CNN - Bi to multi-classifier: Given that the *trueneg* images are easily confused with other classes, a nested CNN with a binary classifier picking up all the *trueneg* images followed by a multi-class classifier for the rest of the images can potentially improve our current model.

Nested CNN - localization prior to classification: Our current model can sometimes be distracted by unimportant information and thus make wrong decisions. Implementing a hand-localization CNN prior to the classification CNN can potentially improve our model by a great extant.

8 Contributions

Veronica Peng: running experiments, construct ensemble, confusion matrix visualization
Xi Yu: dataset preprocessing, construct network prototype, tSNE analysis
Wenxi Zhao: construct metrics, model debugging, error analysis

9 Appendix

9.1 Dataset

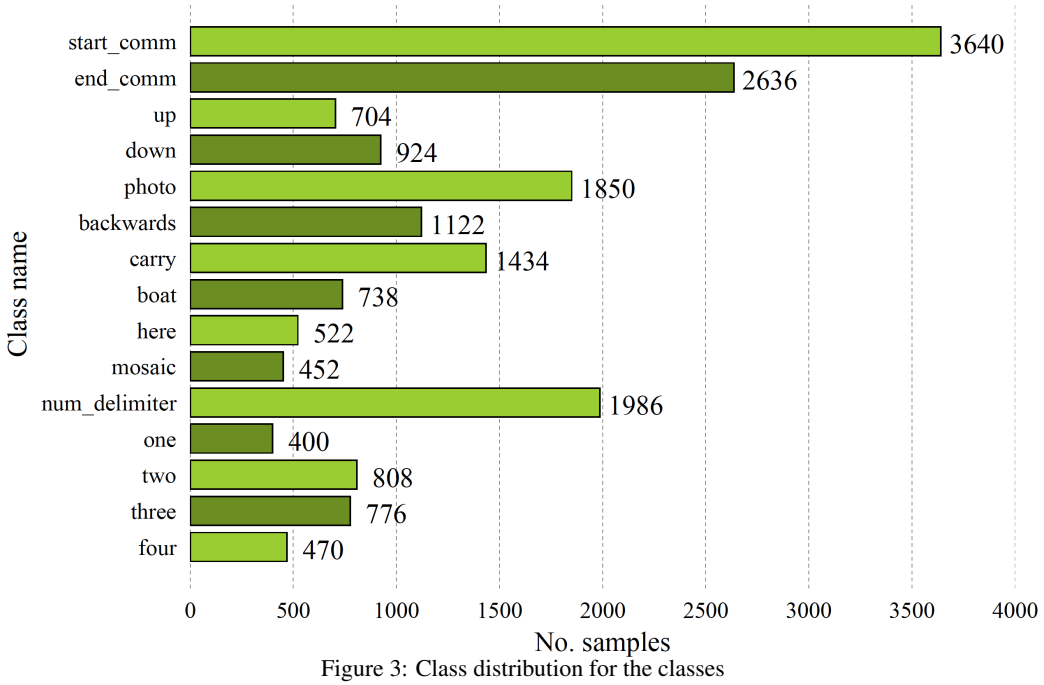


Figure 3: Class distribution for the classes

Figure 4: Sample input

9.2 Experiment 1: resnet18, resnet 50 and resnet101 on all-scenarios

9.2.1 Loss/Accuracy/Learning rate vs. Epoch number

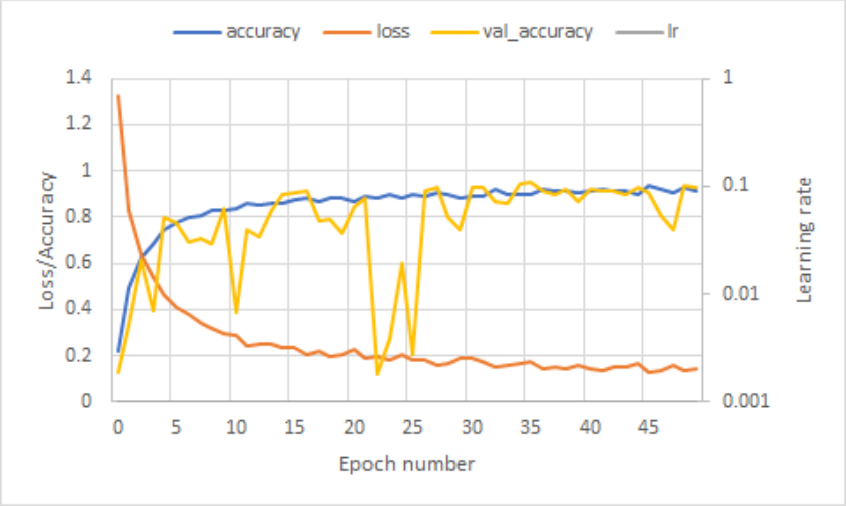


Figure 5: Loss/Accuracy/Learning rate for resnet-18 trained on **all-scenarios**

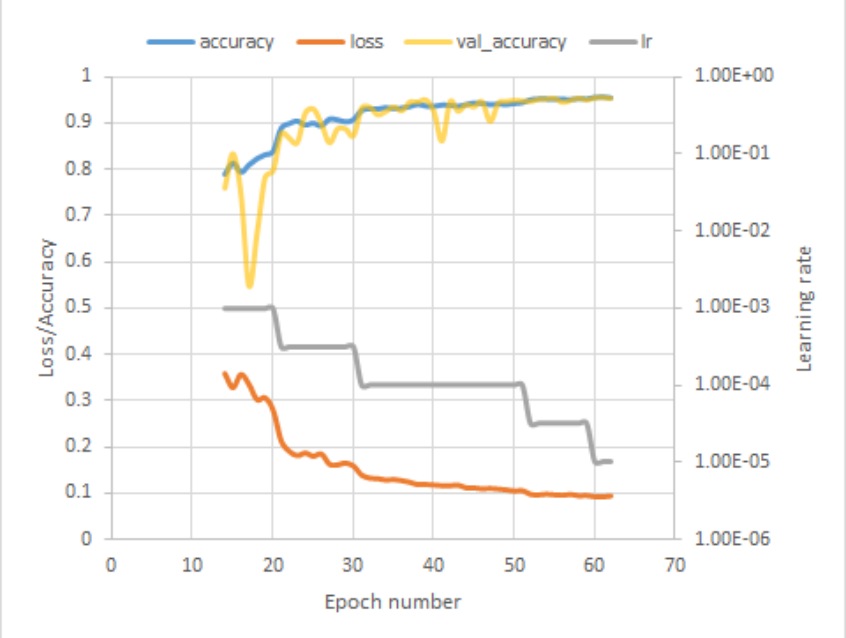


Figure 6: Loss/Accuracy/Learning rate for resnet-50 trained on **all-scenarios**

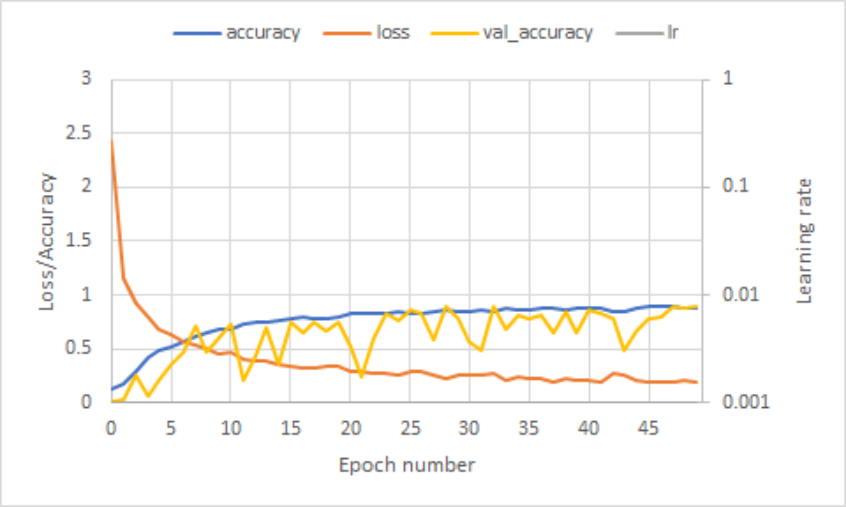


Figure 7: Loss/Accuracy/Learning rate for resnet-101 trained on **all-scenarios**

9.2.2 Confusion Matrix

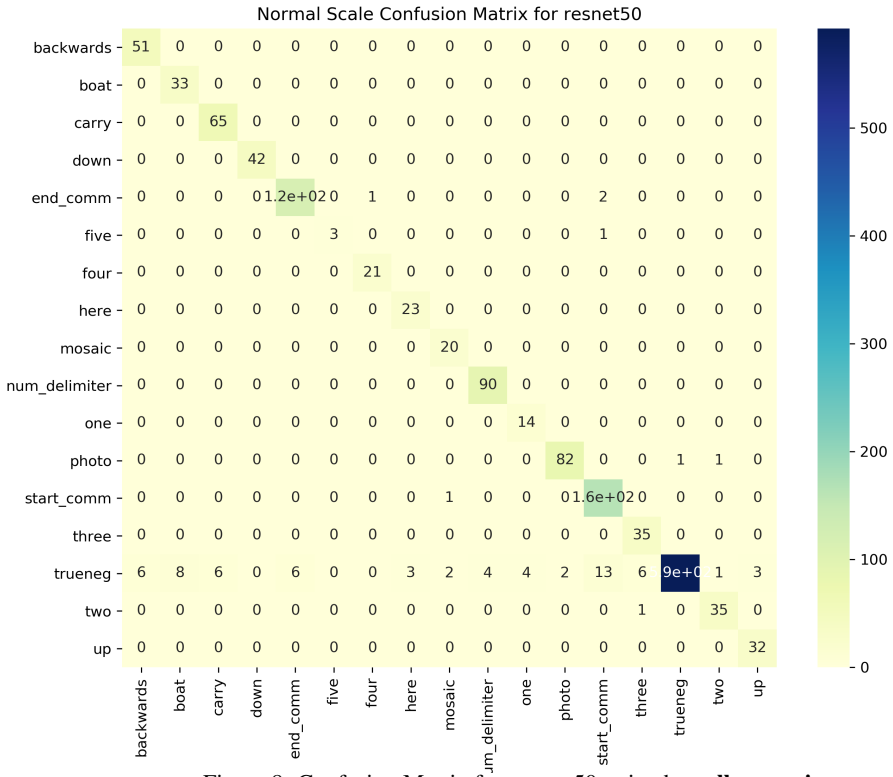


Figure 8: Confusion Matrix for resnet-50 trained on **all-scenarios**

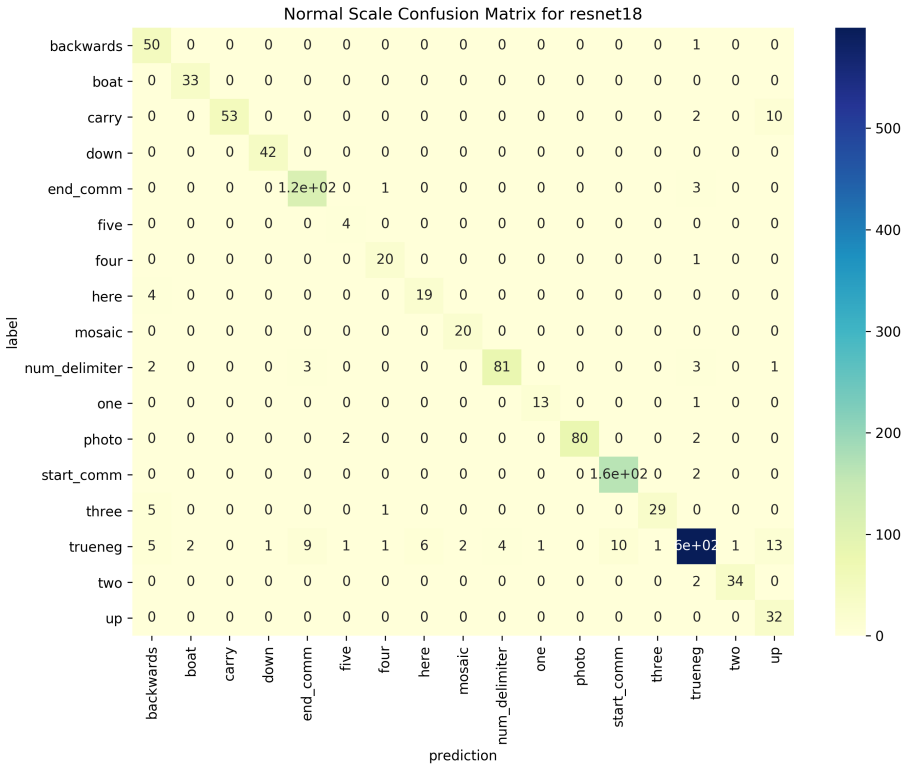


Figure 9: Confusion Matrix for resnet-18 trained on **all-scenarios**

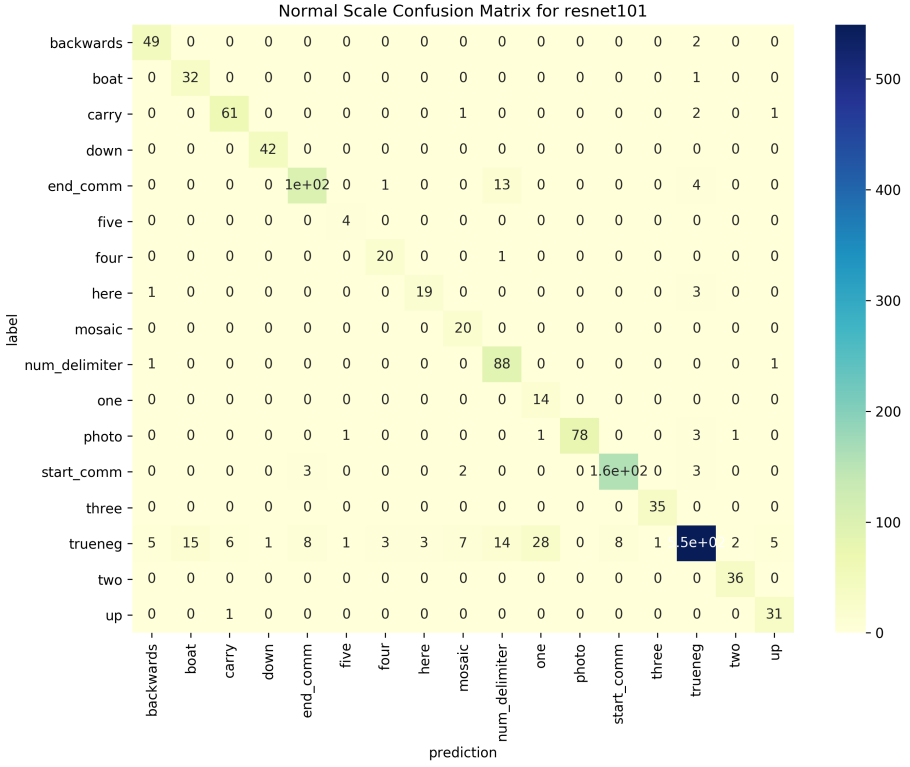


Figure 10: Confusion Matrix for resnet-101 trained on **all-scenarios**

9.2.3 AUC accuracy

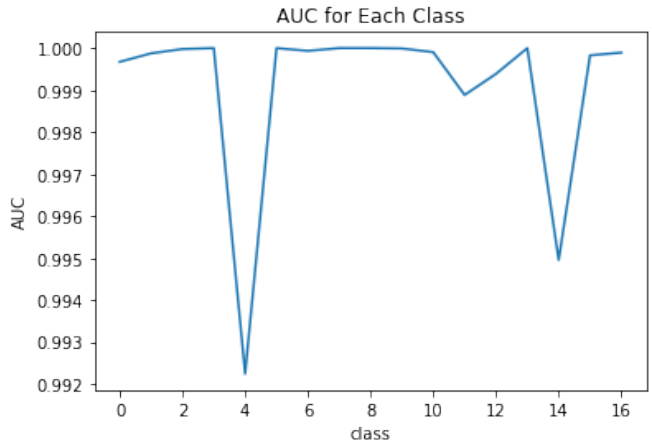


Figure 11: AUC accuracy for resnet-50 trained on **all-scenarios**

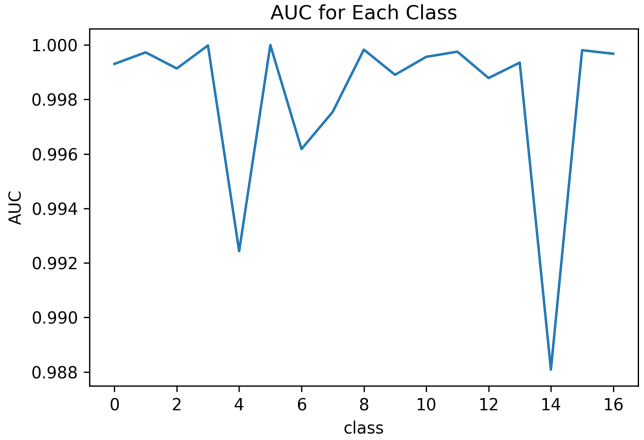


Figure 12: AUC accuracy for resnet-18 trained on **all-scenarios**

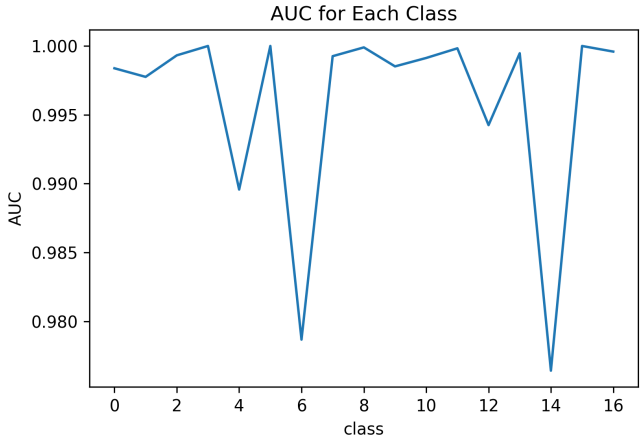


Figure 13: AUC accuracy for resnet101 trained on **all-scenarios**

9.2.4 Precision, Recall and F1 scores for subclasses

```
##### The first score #####
With beta =  2 ,
the tuned f1 score is:  0.9527285443280357

##### The Second score #####
The weighted AUC score is:  0.9970031315851081

##### Classification Report #####
```

	precision	recall	f1-score	support
0	0.895	1.000	0.944	51
1	0.805	1.000	0.892	33
2	0.915	1.000	0.956	65
3	1.000	1.000	1.000	42
4	0.951	0.975	0.963	120
5	1.000	0.750	0.857	4
6	0.955	1.000	0.977	21
7	0.885	1.000	0.939	23
8	0.870	1.000	0.930	20
9	0.957	1.000	0.978	90
10	0.778	1.000	0.875	14
11	0.976	0.976	0.976	84
12	0.912	0.994	0.951	166
13	0.833	1.000	0.909	35
14	0.998	0.902	0.948	656
15	0.946	0.972	0.959	36
16	0.914	1.000	0.955	32
accuracy			0.952	1492
macro avg	0.917	0.975	0.942	1492
weighted avg	0.957	0.952	0.952	1492

Figure 14: Precision, Recall and F1 scores for resnet-50 trained on **all-scenarios**

```
##### The first score #####
With beta =  2 ,
the tuned f1 score is:  0.9330828811260684

##### The Second score #####
The weighted AUC score is:  0.9937449172931255

##### Classification Report #####
```

	precision	recall	f1-score	support
backwards	0.758	0.980	0.855	51
boat	0.943	1.000	0.971	33
carry	1.000	0.815	0.898	65
down	0.977	1.000	0.988	42
end_comm	0.906	0.967	0.935	120
five	0.571	1.000	0.727	4
four	0.870	0.952	0.909	21
here	0.760	0.826	0.792	23
mosaic	0.909	1.000	0.952	20
num_delimiter	0.953	0.900	0.926	90
one	0.929	0.929	0.929	14
photo	1.000	0.952	0.976	84
start_comm	0.943	0.988	0.965	166
three	0.967	0.829	0.892	35
trueneg	0.972	0.913	0.942	656
two	0.971	0.944	0.958	36
up	0.571	1.000	0.727	32
accuracy			0.931	1492
macro avg	0.882	0.941	0.902	1492
weighted avg	0.942	0.931	0.933	1492

Figure 15: Precision, Recall and F1 scores for resnet-18 trained on **all-scenarios**

```
##### The first score #####
With beta =  2 ,
the tuned f1 score is:  0.9011826755780331

##### The Second score #####
The weighted AUC score is:  0.9875748979294726

##### Classification Report #####
```

	precision	recall	f1-score	support
backwards	0.875	0.961	0.916	51
boat	0.681	0.970	0.800	33
carry	0.897	0.938	0.917	65
down	0.977	1.000	0.988	42
end_comm	0.903	0.850	0.876	120
five	0.667	1.000	0.800	4
four	0.833	0.952	0.889	21
here	0.864	0.826	0.844	23
mosaic	0.667	1.000	0.800	20
num_delimiter	0.759	0.978	0.854	90
one	0.326	1.000	0.491	14
photo	1.000	0.929	0.963	84
start_comm	0.952	0.952	0.952	166
three	0.972	1.000	0.986	35
trueneg	0.968	0.837	0.898	656
two	0.923	1.000	0.960	36
up	0.816	0.969	0.886	32
accuracy			0.897	1492
macro avg	0.828	0.951	0.872	1492
weighted avg	0.919	0.897	0.902	1492

Figure 16: Precision, Recall and F1 scores for resnet101 trained on **all-scenarios**

9.3 Experiment 2: resnet18 and resnet 50 on dataset2500

9.3.1 Loss/Accuracy/Learning rate vs. Epoch number

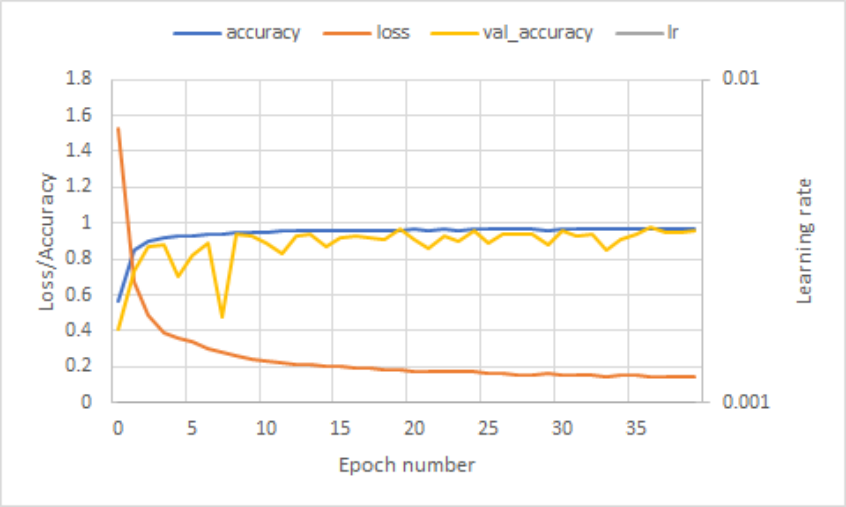


Figure 17: Loss/Accuracy/Learning rate for resnet-18 trained on **dataset2500**

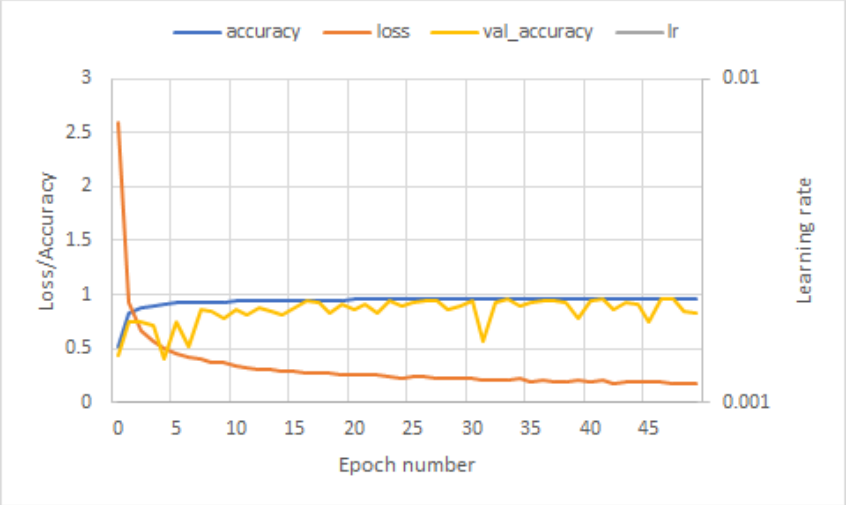


Figure 18: Loss/Accuracy/Learning rate for resnet-50 trained on **dataset2500**

9.3.2 Confusion Matrix

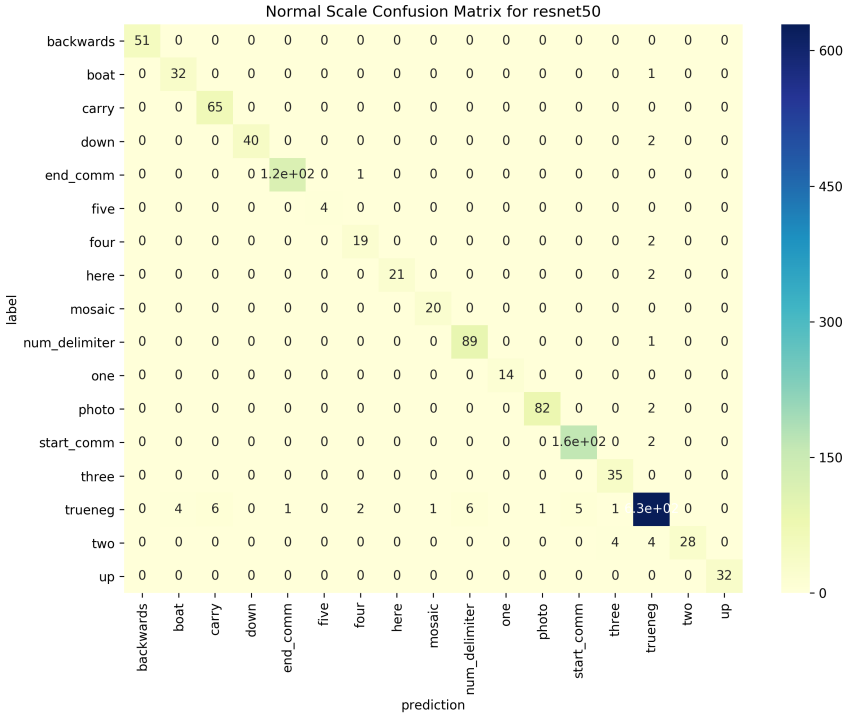


Figure 19: Confusion Matrix for resnet-50 trained on **dataset2500**

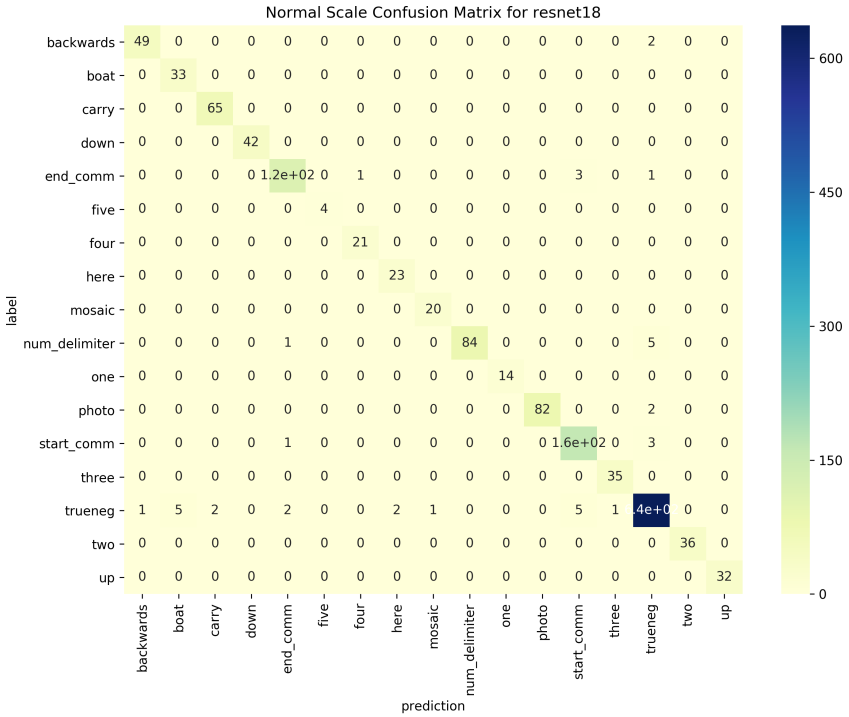


Figure 20: Confusion Matrix for resnet-18 trained on **dataset2500**

9.3.3 AUC accuracy

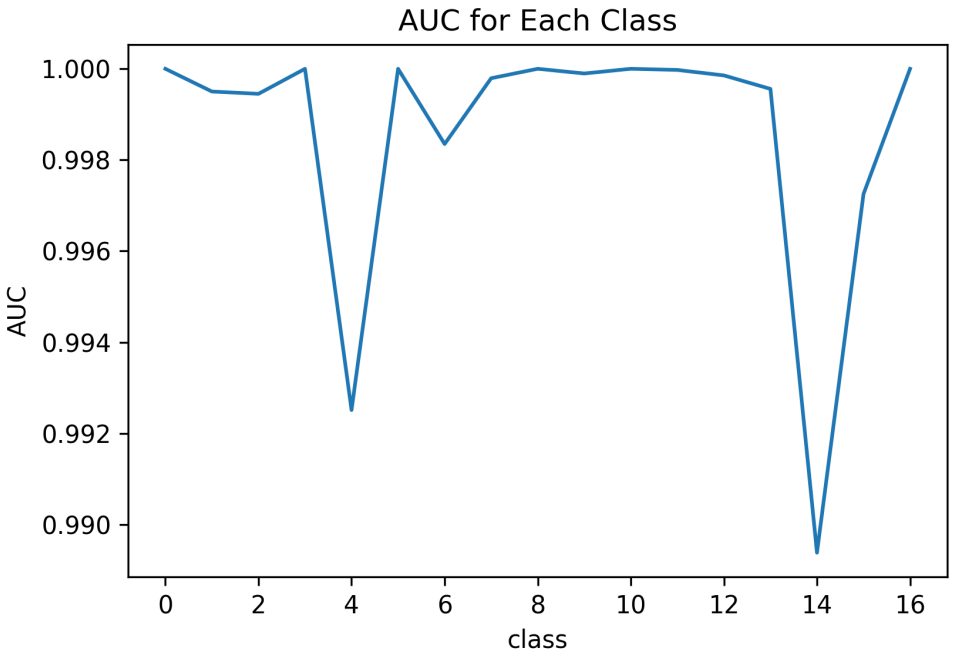


Figure 21: AUC accuracy for resnet-50 trained on **dataset2500**

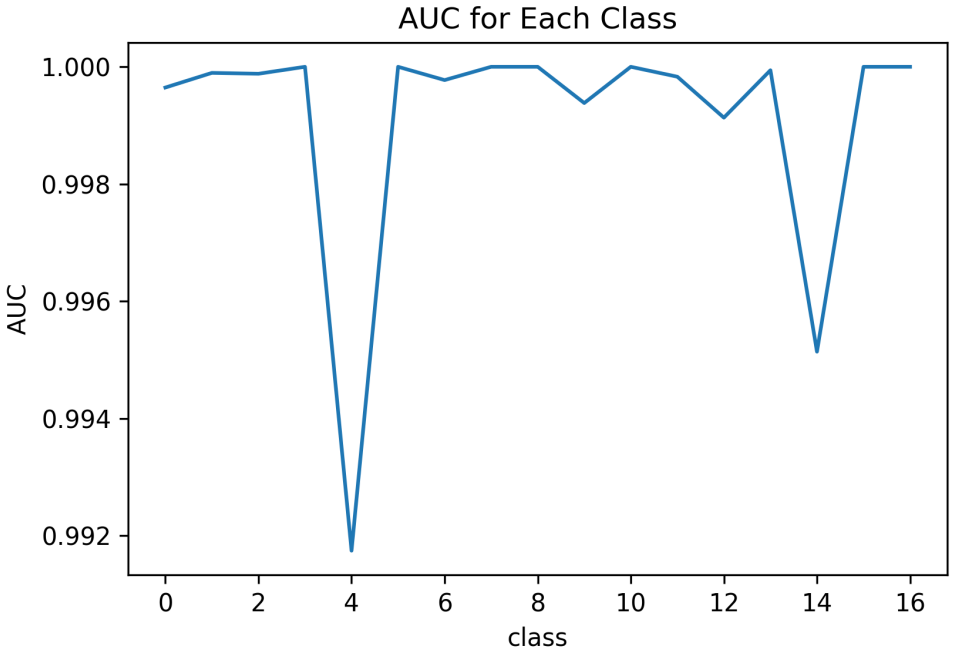


Figure 22: AUC accuracy for resnet-18 trained on **dataset2500**

9.3.4 Precision, Recall and F1 scores for subclasses

```
##### The first score #####
With beta =  2 ,
the tuned f1 score is:  0.9680653159159077

##### The Second score #####
The weighted AUC score is:  0.9945665554505845

##### Classification Report #####
```

	precision	recall	f1-score	support
backwards	1.000	1.000	1.000	51
boat	0.889	0.970	0.928	33
carry	0.915	1.000	0.956	65
down	1.000	0.952	0.976	42
end_comm	0.992	0.992	0.992	120
five	1.000	1.000	1.000	4
four	0.864	0.905	0.884	21
here	1.000	0.913	0.955	23
mosaic	0.952	1.000	0.976	20
num_delimiter	0.937	0.989	0.962	90
one	1.000	1.000	1.000	14
photo	0.988	0.976	0.982	84
start_comm	0.970	0.988	0.979	166
three	0.875	1.000	0.933	35
trueneg	0.975	0.959	0.967	656
two	1.000	0.778	0.875	36
up	1.000	1.000	1.000	32
accuracy			0.968	1492
macro avg	0.962	0.966	0.963	1492
weighted avg	0.969	0.968	0.968	1492

Figure 23: Precision, Recall and F1 scores for resnet-50 trained on **dataset2500**

```
##### The first score #####
With beta =  2 ,
the tuned f1 score is:  0.9746665556409453

##### The Second score #####
The weighted AUC score is:  0.9970320364528669

##### Classification Report #####
```

	precision	recall	f1-score	support
backwards	0.980	0.961	0.970	51
boat	0.868	1.000	0.930	33
carry	0.970	1.000	0.985	65
down	1.000	1.000	1.000	42
end_comm	0.966	0.958	0.962	120
five	1.000	1.000	1.000	4
four	0.955	1.000	0.977	21
here	0.920	1.000	0.958	23
mosaic	0.952	1.000	0.976	20
num_delimiter	1.000	0.933	0.966	90
one	1.000	1.000	1.000	14
photo	1.000	0.976	0.988	84
start_comm	0.953	0.976	0.964	166
three	0.972	1.000	0.986	35
trueneg	0.980	0.971	0.975	656
two	1.000	1.000	1.000	36
up	1.000	1.000	1.000	32
accuracy			0.975	1492
macro avg	0.972	0.987	0.979	1492
weighted avg	0.975	0.975	0.975	1492

Figure 24: Precision, Recall and F1 scores for resnet-18 trained on **dataset2500**

9.4 Experiment 3: resnet18 on dataset2500 with different input image sizes

9.4.1 Loss/Accuracy/Learning rate vs. Epoch number

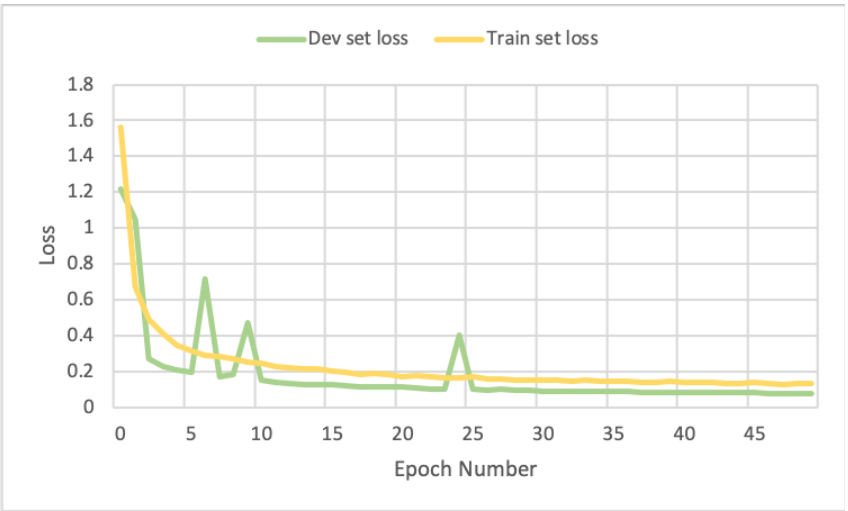


Figure 25: Loss/Accuracy/Learning rate for resnet-18 trained on **dataset2500** with input image size 320×240

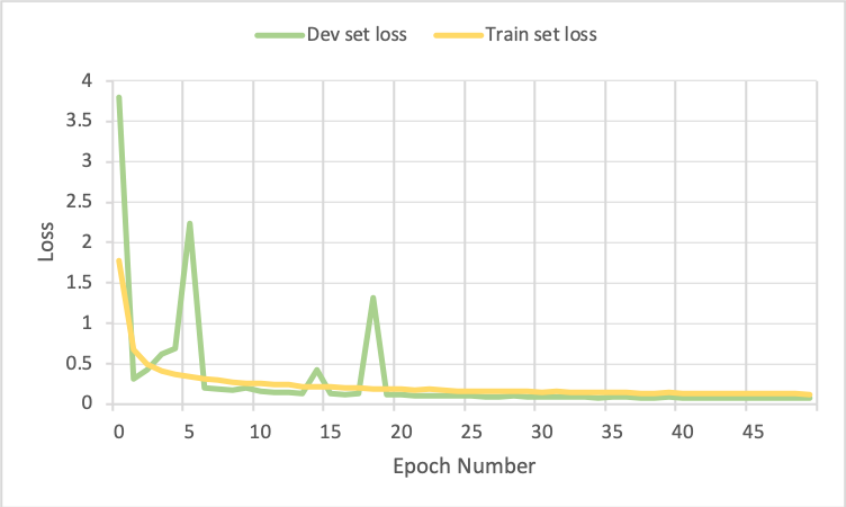


Figure 26: Loss/Accuracy/Learning rate for resnet-18 trained on **dataset2500** with input image size 480×360

9.4.2 Confusion Matrix

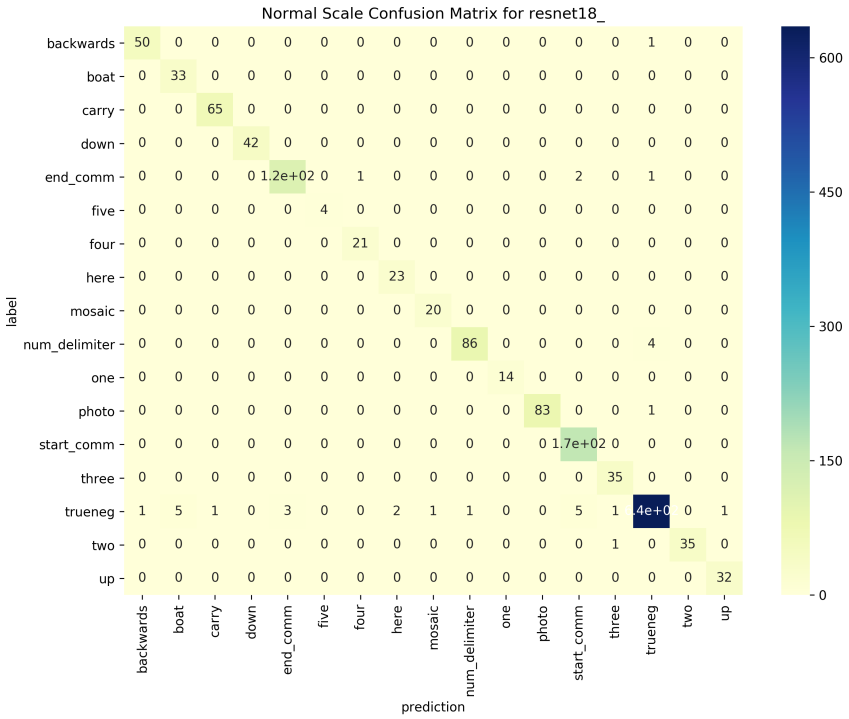


Figure 27: Confusion Matrix for resnet-18 trained on **dataset2500** with input image size 320×240

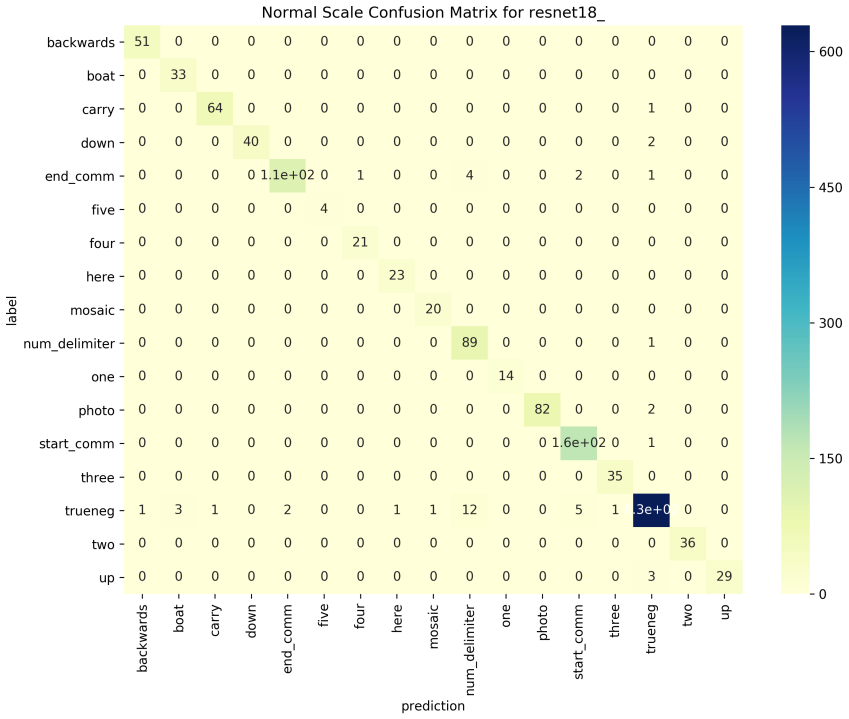


Figure 28: Confusion Matrix for resnet-18 trained on **dataset2500** with input image size 480×360

9.4.3 AUC accuracy

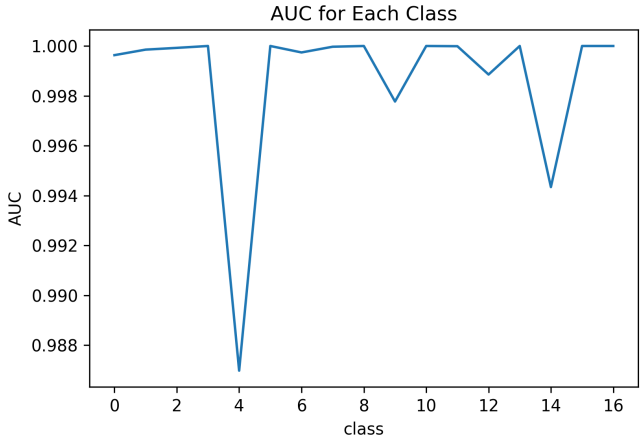


Figure 29: AUC accuracy for resnet-18 trained on **dataset2500** with input image size 320×240

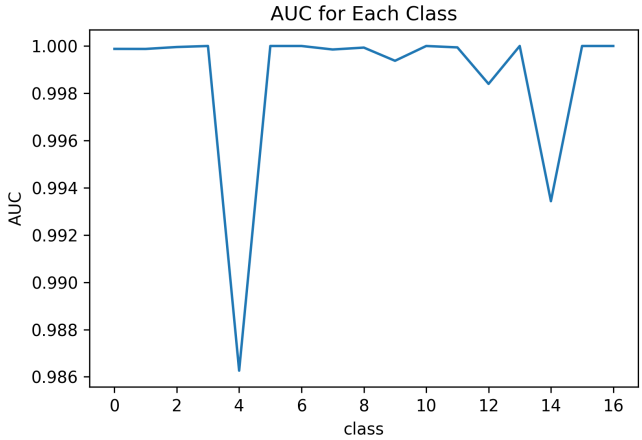


Figure 30: AUC accuracy for resnet-18 trained on **dataset2500** with input image size 480×360

9.4.4 Precision, Recall and F1 scores for subclasses

```
##### The first score #####
With beta =  2 ,
the tuned f1 score is:  0.9787043862186402

##### The Second score #####
The weighted AUC score is:  0.9961795257752977

##### Classification Report #####
```

	precision	recall	f1-score	support
backwards	0.980	0.980	0.980	51
boat	0.868	1.000	0.930	33
carry	0.985	1.000	0.992	65
down	1.000	1.000	1.000	42
end_comm	0.975	0.967	0.971	120
five	1.000	1.000	1.000	4
four	0.955	1.000	0.977	21
here	0.920	1.000	0.958	23
mosaic	0.952	1.000	0.976	20
num_delimiter	0.989	0.956	0.972	90
one	1.000	1.000	1.000	14
photo	1.000	0.988	0.994	84
start_comm	0.960	1.000	0.979	166
three	0.946	1.000	0.972	35
trueneg	0.989	0.968	0.978	656
two	1.000	0.972	0.986	36
up	0.970	1.000	0.985	32
accuracy			0.979	1492
macro avg	0.970	0.990	0.979	1492
weighted avg	0.979	0.979	0.979	1492

Figure 31: Precision, Recall and F1 scores for resnet-18 trained on **dataset2500** with input image size 320×240

```
##### The first score #####
With beta =  2 ,
the tuned f1 score is:  0.9701846792447452

##### The Second score #####
The weighted AUC score is:  0.9957756314655417

##### Classification Report #####
```

	precision	recall	f1-score	support
backwards	0.981	1.000	0.990	51
boat	0.917	1.000	0.957	33
carry	0.985	0.985	0.985	65
down	1.000	0.952	0.976	42
end_comm	0.982	0.933	0.957	120
five	1.000	1.000	1.000	4
four	0.955	1.000	0.977	21
here	0.958	1.000	0.979	23
mosaic	0.952	1.000	0.976	20
num_delimiter	0.848	0.989	0.913	90
one	1.000	1.000	1.000	14
photo	1.000	0.976	0.988	84
start_comm	0.959	0.994	0.976	166
three	0.972	1.000	0.986	35
trueneg	0.983	0.959	0.971	656
two	1.000	1.000	1.000	36
up	1.000	0.906	0.951	32
accuracy			0.970	1492
macro avg	0.970	0.982	0.975	1492
weighted avg	0.972	0.970	0.970	1492

Figure 32: Precision, Recall and F1 scores for resnet-18 trained on **dataset2500** with input image size 480×360

9.5 Error Analysis

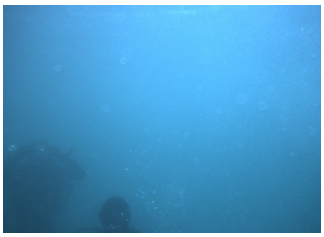
Mis-classified Photo	True Label	Prediction	Categories of Error
	trueneg	start_com	**Similar gesture:** The gesture is unintentionally similar to "start communication". To this point, its hard for the model to tell, more data in start_com class may help.
	trueneg	start_com	**Dim light:** Dim light made the gesture unclear and hard for the model to learn. More training photo taken in this scenario should be fed into the model.
	trueneg	start_com	**Similar body figure:** Instead of feeding the whole body image into the model, the model can be improved by first using a CNN to localize the arm-hand segment and then feed the segment data into the classifier.
	trueneg	start_com	**Fuzziness:** Might be solved by adding median filters during the image pre-processing.
	trueneg	end_com	**More than one diver:** More training photo with more than one divers needed for improve the model.
	trueneg	start_com	**Distraction:** The fluorescent tube in front of the diver's chest might be misrecognized as the gloves that highlight hands. More data needed for the model to learn their difference.

Table 6: Error Analysis

9.6 Metrics

Primary Metrics: Accuracy $= \frac{TP+TN}{TP+FP+FN+TN}$

Other metrics: Weighted AUC $= \sum_{i=1}^{c} AUC_i \times \frac{|C_i|}{|C_{total}|}$

Other metrics: Weighted Recall(WR) $= \sum_{i=1}^{c} (\frac{TP}{TP+FN})_i \times \frac{|C_i|}{|C_{total}|}$

Other metrics: Weighted Precision(WP) $= \sum_{i=1}^{c} (\frac{TP}{TP+FP})_i \times \frac{|C_i|}{|C_{total}|}$

Other metrics: Tuned F1 $= \frac{(\beta^2+1)\times WR\times WP}{WR+\beta\times WP}$ (we set $\beta = 2$ for the emphasis on weighted recall)

References

[Dataset] Gomez Chavez A., Ranieri A., Chiarella D., Zereik E., Babic A. & Birk A. (2018) CADDY Underwater Stereo-Vision Dataset for Human–Robot Interaction (HRI) in the Context of Diver Activities. *Journal of Marine Science and Engineering* **7**(1): 16.

[1] Chiarella, D., Bibuli, M., Bruzzone, G., Caccia, M., Ranieri, A., Zereik, E., ... & Cutugno, P. (2018) A novel gesture-based language for underwater human–robot interaction. *Journal of Marine Science and Engineering* **6**(3), 91.

[2] Wang, H., Kläser, A., Schmid, C., & Liu, C.L. (2013) Dense trajectories and motion boundary descriptors for action recognition. *International journal of computer vision*, **103(1)**, pp.60-79.

[3] Yang, M. H., & Ahuja, N. (2001) Recognizing hand gestures using motion trajectories. *In Face Detection and Gesture Recognition for Human-Computer Interaction* (pp. 53-81). Springer, Boston, MA.

[4] Shotton, J., Fitzgibbon, A., Cook, M., Sharp, T., Finocchio, M., Moore, R., ... & Blake, A. (2011) Real-time human pose recognition in parts from single depth images. In *CVPR 2011* (pp. 1297-1304). Ieee.

[5] Keskin, C., Kıraç, F., Kara, Y. E., & Akarun, L. (2013) Real time hand pose estimation using depth sensors. In *Consumer depth cameras for computer vision* (pp. 119-137). Springer, London.

[6] Neverova, N., Wolf, C., Taylor, G., & Nebout, F. (2015) Moddrop: adaptive multi-modal gesture recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **38**(8), 1692-1706.

[7] Varma, R. (n.d.). Picking Loss Functions - A comparison between MSE, Cross Entropy, and Hinge Loss. Retrieved from https://rohanvarma.me/Loss-Functions/

[Library: NumPy] Travis E. Oliphant (2006) A guide to NumPy, USA: Trelgol Publishing. Stéfan van der Walt, S. Chris Colbert and Gaël Varoquaux (2011) The NumPy Array: A Structure for Efficient Numerical Computation, Computing in Science Engineering, 13, 22-30, DOI:10.1109/MCSE.2011.37

[Library: Matplotlib] John D. Hunter (2007) Matplotlib: A 2D Graphics Environment, Computing in Science Engineering, 9, 90-95, DOI:10.1109/MCSE.2007.55

[Library: pandas] Wes McKinney (2010) Data Structures for Statistical Computing in Python, Proceedings of the 9th Python in Science Conference, 51-56

[Library: scikit-learn] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, Édouard Duchesnay (2011) Scikit-learn: Machine Learning in Python, Journal of Machine Learning Research, 12, 2825-2830

[Library: seaborn] DOI:doi.org/10.5281/zenodo.54844)

[Library: Cython] Stefan Behnel, Robert Bradshaw, Craig Citro, Lisandro Dalcin, Dag Sverre Seljebotn and Kurt Smith. Cython: The Best of Both Worlds, Computing in Science and Engineering, 13, 31-39 (2011), DOI:10.1109/MCSE.2010.118

[Library: keras] Chollet, François and others (2015) Keras. Retrived from https://keras.io