2019

NuNet: A Deep Learning Approach for U.S. Traffic Sign Recognition

Emmanuel Borkor Nuakoh
North Carolina Agricultural and Technical State University

NuNet: A Deep Learning Approach for U.S. Traffic Sign Recognition

Emmanuel Borkor Nuakoh

North Carolina A&T State University

A dissertation submitted to the graduate faculty

in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Department: Computer Science

Major: Computer Science

Major Professor: Dr. Kaushik Roy

Greensboro, North Carolina

2019

The Graduate College
North Carolina Agricultural and Technical State University

This is to certify that the Doctoral Dissertation of

Emmanuel Borkor Nuakoh

has met the dissertation requirements of
North Carolina Agricultural and Technical State University

Greensboro, North Carolina
2019

Approved by:

Dr. Kaushik Roy Major Professor	Dr. Jinsheng Xu Committee Member
Dr. Xiaohong Yuan Committee Member & Department Chair	Dr. Albert Esterline Committee Member
Dr. Evelyn Sowells-Boone Committee Member	Dr. Clay S. Gloster, Jr. Interim Dean, The Graduate College

Biographical Sketch

Emmanuel Borkor Nuakoh was born in Bibiani in the Western Region of Ghana. He completed Ghana National College in 2004 after which he entered the University of Mines and Technology (UMAT) to pursue his Bachelor of Science in Geological Engineering in 2007. After receiving his Bachelor's degree in 2011, Emmanuel worked in the mining industry in Ghana, Burkina Faso, and Niger as an exploration geologist before starting his master's program in 2013. In December of 2014, he received his master's degree in Computer Science from North Carolina Agricultural and Technical State University (NCAT) and is due to receive his doctorate in December 2019 at the same institution.

Whiles pursuing his Ph.D., Emmanuel got introduced into entrepreneurship and founded SoftOffice Inc. in September of 2018. He developed a passion for entrepreneurship after being accepted into the National Science Foundation Innovation-Corps (NSF I-Corps) program at the University of North Carolina at Greensboro (UNCG); which ran concurrently with a 5-week accelerator program on NCAT campus called Aggie Accelerator. After successful completion from the I-Corps node program, he was accepted into the National I-Corps program called the I-Corps Teams – National Innovation Network (NIN) to perform customer discovery.

Dedication

This work is dedicated to my mom Mrs. Comfort Nkrumah for her support, prayers, and guidance throughout my life, my dad, Mr. Philip Nuako, for his support and encouragement, my siblings, Kyei, Ebenezer, Kwabena, Evelyn and Serwaa for their encouragements and moral support, and to my daughter Ama Nkrumah. Love you and God bless you!

Acknowledgments

I would like to acknowledge God for His protection and guidance. A special thank you to my professor, Dr. Kaushik Roy for his patience and guidance. I would also like to send a special thanks to my dissertation committee members, Dr. Albert Esterline, Dr. Jinsheng Xu and Dr. Dorothy Yuan for your unwavering support towards the completion of this work. Last but not least, thanks to Kelly Morgan, director of research communications for her support during this journey.

Table of Contents

List of Figures

List of Tables

Abstract

Traffic Sign Recognition System (TSRS) is an Advanced Driver Assistance System (ADAS) that helps drivers with perception to ensure road safety. Two main activities are performed in TSRS: detection and classification. The detection aspect involves localizing traffic signs in an image frame while the classification aspect deals with recognizing the class of the detected sign. Research in this area has mainly focused on German, Belgium, Sweden, Chinese, and several other datasets using different approaches. However, limited research has been conducted using U.S. traffic signs; the ones that have been conducted are mostly concerned with speed limit signs recognition. This work expands the classification of U.S. traffic signs to cover all the publicly available classes.

Convolutional Neural Networks (CNN) have shown a lot of success on European datasets. One key issue with CNN is that it requires a lot of data for training. This research introduces a new model, called NuNet, with a new dataset, CIB TS V1. The model is used to classify the CIB dataset and LISA benchmark (Møgelmose et al., 2012). Results from running NuNet on LISA and CIB are then compared with those of a modified VGGNet. The new model trains and converges faster than VGGNet and is adaptable to both large and sparse datasets. Experiments conducted with the VGGNet show training and validation accuracies of 99.93% and 99.83, respectively on the LISA dataset. However, it overfits on the CIB dataset with training and validation accuracies of 100% and 96.92%, respectively. This is because the deep net cannot generalize well on small datasets and thereby learns noise. NuNet, on the other hand, generalizes well on smaller datasets recording accuracies of 99.73% and 99.83% on LISA for the training and validation sets respectively, and 100% for both training and validation sets on the CIB dataset. NuNet trains for 4 hours on LISA and an hour on CIB whereas VGGNet trains for 23 hours on LISA and 8 hours on CIB.

CHAPTER 1

Introduction

Traffic sign recognition systems (TSRS) consist of two activities; detecting and recognizing traffic signs (Escalera, A. d. l., et al 1997). Detecting the sign deals with localizing a traffic sign in an image frame with background noise using shape, color or form. Recognition of a detected sign has to do with classifying the detected sign into a given class of traffic signs. This work focuses on the recognition aspect of the TSRS. The U.S. traffic sign has received little research attention compared to European traffic signs. Most of the research that has been conducted with U.S. traffic signs have focused mainly on speed limit signs or broader categories such as warning signs, prohibitory signs or speed limit signs. This work expands the boundaries of previous research to classify all traffic signs in the publicly available LISA dataset (Møgelmose et al., 2012). A new dataset is also introduced for the purpose of this work.

1.1 Background and Motivation

Research that has been conducted in the area of traffic sign recognition has mostly focused on using European traffic signs, such as the German Traffic Sign Recognition Benchmark (GTSRB) (Stallkamp et al., 2011) and the Belgium Traffic Sign Classification (BTSC) (Timofte et al., 2011) datasets. However, limited research has been conducted on US traffic signs (Li, Y., et al. 2016). Most research on U.S. traffic signs relies on speed signs recognition or broader categories of traffic signs such as speed signs, prohibitory signs, and warning signs. This research offers a more granular classification of U.S. traffic signs to cover publicly available classes including the LISA TS, LISA Extension benchmarks, and a newly introduced dataset, CIB TS V1 (Center for Biometrics Traffic Sign Volume 1) benchmark.

Several methods have been successfully applied in TSRS and shown outstanding results. Convolutional Neural Networks (CNN) is one of the methods that has yielded high success in classifying traffic signs. This deep learning methodology has been tested exhaustively on European traffic signs; winning awards in the GTSRB competition (Stallkamp et al., 2011). However, little research has been conducted using deep learning approaches to classify U.S. traffic signs. This research proposes a CNN model to classify U.S. traffic signs.

The research first adopted a publicly available deep learning model, the VGGNet, to classify LISA TS benchmark, a publicly available U.S. traffic signs dataset. VGGNet is a deep CNN with 16 layers. It was developed by the Visual Geometry Group (VGG) from the University of Oxford, (Simonyan & Zisserman, 2015). It recorded an error of 7.5% during validation and 7.4% in testing on GTSRB. The model is improved with the multi-stage architecture proposed by (Sermanet & LeCun 2011), which helps VGGNet to generalize well with the dense network over multiple scales (Sermanet et al., 2013a). The VGGNet 16 used in this research adopts a multi-stage architecture as shown in Figure 1.1.

Sermanet & LeCun (2011) introduced a CNN architecture that used multi-scale features to feed the output of the first stage after pooling operation of the second stage as shown in Figure 1.2. The application of a second subsampling with the output from the first stage was reported to increase the accuracy of the network. This work is an application of a combination of their research methodology (Simonyan & Zisserman, 2015) to classify the U.S. traffic sign dataset. This was to develop a deep neural network to classify not only U.S. speed limit signs but extended to other U.S. traffic signs as contained in the LISA TS dataset. A modified version of the GitHub code[1]

used in the initial experiment was reported by Nuakoh et. al., (2019) and formed an important motivation for the success of this research.

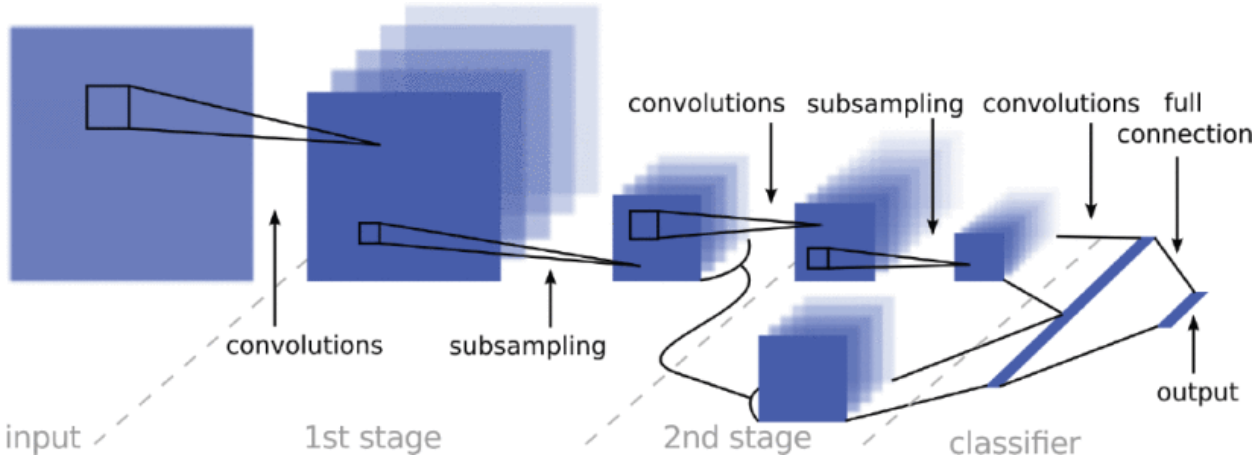

Figure 1.1. Multi-Stage Convolution Neural Network Architecture

1.2 Problem Statement and Hypothesis

The application of traffic sign recognition systems in ADAS has seen some commercial success with limited functionality (Jurišić et al., 2015). The limitation is not only linked to the number of supported traffic signs, but also the areas of the road network where they are most effective. This leaves the problem half-solved and to fully solve it, a system that is robust for all-weather, lighting and physical sign conditions including occlusions and accommodating different datasets must be developed. Whereas this is a generally big problem, it can be tackled discretely; starting first with extending the number of signs current systems support.

The Vienna Convention on Road Signs and Signals offers a broad category of traffic signs. This coupled with several modifications for individual countries makes it difficult to develop a multi-purpose TSRS that is adaptable in different countries. Also, the lack of standardized datasets for training and testing models poses another problem in this area of research, making it quite difficult to compare the performance of new models with old ones.

Unbalanced traffic sign datasets also present another layer of difficulty for developing TSR systems using deep learning. Machine learning depends on lots of data for training the algorithm to perform better on new data, however, most traffic sign datasets are porous and need to be augmented in other ways to balance them. While data augmentation offers some improvement, it is rather imperative to have a dataset that comes somehow already balanced, for real-world applications.

The research hypotheses are as follows:

- Deep neural networks with fewer layers train faster than ones with many layers, however, they do not generalize well.
- Increasing the number of layers improves the model performance on sparse datasets if the increase can be delayed until model performance stalls.
- The proposed architecture with fewer layers trains faster than VGGNet.

1.3 Research Questions

- Is there a way to improve the performance of a deep learning model while improving the speed at the same time?
- Can a deep learning architecture be developed that performs well on sparse datasets?

1.4 Contribution

The contributions of this research are three-fold:

- The main contribution of this work is the development of a new deep learning model architecture (NuNet) that was used to classify U.S. traffic signs presented in the LISA benchmark.
- The research uses VGGNet, an already existing model on the LISA dataset to compare the performance of both models as it pertains to accuracy, error/loss, and speed during training.

- The research also introduces a new dataset, the CIB-TS V1 for advancing research in this area.

Limited work has been done on U.S. traffic sign data and no known work has been done using a deep learning technique for classifying all the classes contained in the LISA dataset. The work is done by Li et. Al. (2016) only focused on speed limit signs. This research extends that work to cover all signs with adequate data points for training and validation.

CHAPTER 2

Literature Review

Traffic sign recognition research has gained a lot of traction in recent years. Various methods with wide applicability for image recognition have proven that they can be used to recognize traffic signs as well (Mathias et al., 2013).

Several publicly available datasets have been used for the traffic sign classification problem, including but not limited to: The Belgian Traffic Sign Classification (BTSC) dataset (Timofte et al., 2011), the German Traffic Sign Recognition and Detection Benchmark (GTSRB and GTSDB) (Stallkamp et al., 2011), the Croatian traffic sign dataset (rMASTIF) (Jurišić et al., 2015), the Dataset of Italian Traffic Signs (DITS) (Youssef et al., 2016) and the Tsinghua-Tencent 100 K Chinese benchmark (Zhu et al., 2016). Research into TSRS was boosted since several of these datasets are commonly used to evaluate the performance of computer vision algorithms for traffic sign detection and recognition (Álvaro Arcos-García, et al. 2018)

2.1 Background and Overview of Deep Learning

Deep learning for image recognition has seen wide applicability from biometrics, medical diagnostics, text recognition, speech recognition, and traffic sign recognition just to name a few. According to LeCun et al. (2015) conventional machine learning had a limitation with processing raw data. Deep learning approaches are representation-learning based classifications; with each layer of representation representing the presence or absence of parts of objects such as edges and motif. Subsequent layers detect objects as combinations of identified parts. Unlike conventional machine learning, the layers of deep learning are not designed by humans, but by a general-purpose procedure.

In his technical report, Schmidhuber (2015) suggested that the Neocognitron (Fukushima, 1979, 1980, 2013a) was perhaps the first artificial neural network that deserved rthe attribute, deep. It introduced convolutional neural networks that used Winner-Takes-All-based unsupervised learning rules and spatial averaging for downsampling. (LeCun et al., 1998) would later apply Back Propagation to a Neocognitron-like CNN with adaptive connections. Max-Pooling is used to speed up processing. Deep Neural Networks have been adopted to win many awards including (Cireşan et al. 2011).

DNNs have been adopted to improve perception in autonomous vehicles as well as driver assistance systems. Areas of application include vehicle and lane detections (Huval et al., 2015), lane detection (Li et al., 2017), and pedestrian detection (Luo et al., 2014; Ouyang & Wang, 2012; Ouyang & Wang 2013; Sermanet et al., 2013b; Tian et al. 2015).

2.2 Traffic Sign Recognition with Deep Learning

Torresen et al. (2004) classified traffic speed limit signs based on the first digit of the sign. They used a digit classification system that used a bit array of "1" or "0" to denote a number or otherwise. The bit array assignment was based on how much black or white pixels were encountered in the image. The more the black, a bit array of 1 was assigned while the more the white, a bit array of 0 was assigned. The classified numbers represented by the bit array were further fed to a feed-forward Neural Network trained with a back-propagation algorithm in the final recognition stage. They tested their algorithm on Norwegian speed limit signs and achieved an accuracy of 91% on 198 images.

Moutarde et al. (2007) presented an integrated system for speed limit signs detection, tracking, and recognition of European and U.S. speed limit signs with a similar approach as Torresen et al. (2004) but improved to capture the second and third digits to cater for signs that do

not end with “0” and 3-digit speed limits such as “110”. The work conducted by Torrensen et al. (2004) was based on Norwegian speed limit signs, which only end with a “0”, but some U.S. traffic signs end with a “5”. Each detected digit is normalized and fed into a multilayer perceptron neural network optical digit recognition (ODR) module with 10 outputs, each output represents a digit. The recognized signs are then assigned a confidence evaluation in a final step. Their work showed a performance of 89% and 90% for U.S. and European speed limit signs respectively on 281 traffic signs.

Fang et al. (2004) in their work presented an automatic road-sign detection and recognition system that uses spatiotemporal attentional (STA) neural network, configurable adaptive resonance theory (CART) neural networks for classification, and configurable heteroassociative memory (CHAM) neural network for recognition of signs. Their system consists of a sensory component; 2) a perceptual component using STA and long-term memory (LTM); and 3) conceptual analyzers using CART and CHAM. The sensory analyzer extracts the temporal and spatial information from the video sequence. The extracted information then serves as the input stimuli to the STA neural network in the perceptual analyzer. If the stimulation continues, LTM is used to preserve the extracted features of interest. The extracted features from the LTM are fed to CART neural networks and CHAM neural networks in the final stage as conceptual analyzers to respectively classify and recognize signs. The system performed at 85% accuracy in the recognition stage on Chinese traffic signs.

Stallkamp et al. (2011) present the winning algorithms of the “The German Traffic Sign Recognition Benchmark” (GTSRB). The competition was based on multi-class traffic signs and the top 2 winning teams developed ConvNets to reach very high accuracy. *Team IDSA* (Cireşan et al. 2011), the winning team developed a committee of convolutional neural networks (CNN) and

multilayer perceptron (MLP) that were trained on HOG features (HOG3). Their input image was 48X48 and all CNNs had seven hidden layers with the output layer having 43 neurons, each mapping to a class. They trained the best architecture initialized with uniformly random distribution weights with a hyperbolic tangent activation function. The classification had a combined classification rate (CCR) of 98.98%, outperforming humans in some instances. *Team Sermanet* (Sermanet & LeCun 2011) placed second with 98.97% accuracy using a multi-layer ConvNets with more sophisticated non-linearities such as rectified sigmoid, subtractive and divisive local normalizations instead of the traditional hyperbolic tangent sigmoid function. This was to enforce competition between neighboring features.

Stallkamp et al. (2012) report IDSA (Cireşan et al. 2012), again won the 2012 GTSRB competition with a committee of 25 CNN's by using data augmentation and jittering with an accuracy of 99.46%. This time again, *Sermanet* (Sermanet et al. 2012) placed second using a multiscale CNN with an accuracy of 98.31%.

Laguna et al. (2014) present technology for recognizing traffic signs using a Laplacian of Gaussian (LOG) filter to detect edges of regions of interest in greyscale images, comparing detected ROIs with shape patterns and then feeding it to a cross-correlation algorithm for classification. They reported an accuracy of 91.07% detection on 200 images. It is not clear where the data was collected, but their work comes from South Africa, so they might supposedly have obtained it on a South African road network. They had an accuracy of 88.24% when detecting octagonal shapes because octagons were mostly misclassified as circles.

Jin et al. (2014) proposed a hinge loss of stochastic gradient descent (HLSGD) cost function method for training CNNs. This function is similar to support vector machines (SVM) hinge loss and performed faster than the SGD, which is preferred for training CNN. They achieved

an accuracy of 99.65% beating (Ciresan et al., 2012) during testing and a decreased error rate of 35.19%. They reported that hinge loss allows CNN to focus on correctly classifying misclassified training data, unlike cross-entropy cost that tries to make a correct classification more “correct”. Also, unlike SGD, where each iteration must go through costly forward and backward propagation, hinge loss only allows useful training samples to go through backpropagation. That is, if a useless example is encountered, iteration is stopped, and backpropagation is aborted. This makes hinge loss result in a more stable convergence, as most training examples become useless as the model converges and only a few have an effect on the updates of parameters. This decreases the frequency of updates happening and gradually stops over time or rarely happens. Useless examples are omitted from forward propagation and rechecked after several traversals. In doing so, convergence is faster and training speed is improved. This method was tested with an ensemble of 20 CNN's following work done by Ciresan et al. (2012).

Jurišić et al. (2015) developed OneCNN, a convolutional neural network inspired by Sermanet and LeCun (2011). They refrained from using a committee of neural networks for classification of traffic signs, but rather developed a single network that is deeper and more complex but less computationally costly, and used it to classify multiple datasets, particularly, GTSRB, BTSC, and rMASTIF. They achieved an accuracy of 99.11% as against the state-of-the-art (Jin et al., 2014) of 99.65% for the GTSRB; 98.17% for BTSC against the state-of-the-art (Zhu et al.) of 98.77%, and 99.53% for rMASTIF. Their work introduced the rMASTIF dataset. Like this research, their work was only concerned with the classification aspect of the traffic sign recognition system.

Zhu et al. (2016) introduced the Chinese traffic signs dataset called the TT100K benchmark. They divide the dataset into three categories based on their sizes as small, medium

and large. This was to evaluate how the model detects images of different sizes. They use an algorithm with two trained CNNs; one for detection of traffic signs and the other for the classification of these traffic signs. Their algorithm outperformed Fast R-CNN (Girshick, 2015), they report that Fast R-CNN has better performance for recognizing larger objects, however, their model performs at an accuracy of 88% compared to Fast R-CNN's accuracy of 50% on TT100K.

Shustanov & Yakimov (2017) designed an end-to-end CNN algorithm for recognizing traffic signs in real-time. Their system uses the speed of the vehicle to scale the exact coordinates of the traffic signs in subsequent frames, thus improving the detection accuracy, while maintaining the computational cost. They further described how to design a CNN. They developed CNN of 7 convolutional layers, 2 fully connected layers, and a softmax and obtained an accuracy of over 90% on the GTSRB. The algorithm had a high overhead due to the number of layers. This made them reduce it to only one convolutional lay, one fully connected layer and one softmax. This algorithm performed worse than the first and led them to design one with three convolutional layers, one fully connected layer, and one softmax. The model achieved an accuracy of 99.94% for localization and detection of prohibitory and danger traffic signs, but this accuracy is not state-of-the-art for the classification stage.

Yang et al. (2017) proposed a network called deep detection network for real-time traffic sign recognition. Their method was an update to Faster R-CNN and outperformed Faster R-CNN on all levels from small to medium to large traffic signs. The network is composed of four modules: the first module is composed of CNN layers for computing features; the second module which runs parallel with the first layer, hence saving computational cost, is an attention network (AN) which maps attention to coarse-grained regions of interest; the third module is a fully convolutional network which generates a final region proposal from coarse-to-fine grained candidates and; the

final module, a Fast-R-CNN detects and classifies the small targets candidates resulting from the third module in the final recognition stage. They tested their model on the Chinese dataset, TT100K Benchmark, and BTSD, and achieved an accuracy of 80.31% and 94.95% respectively on both benchmarks. Faster R-CNN has an accuracy of 70.63% and 87.07% on the two benchmarks respectively.

Arcos-García et al. (2018) designed a CNN that included Spatial Transformer Networks (STN) and beat work reported by Stallkamp et al. (2011) with the German Traffic Sign Recognition Benchmark (GTSRB) competition. Their proposed model performed at an accuracy of 99.71% at the 21st epoch with three spatial and SGD without momentum as the loss function optimizer on GTSRB. The model beat HLSGD (20 CNN ensemble) (Jin et al., 2014), MCDNN (25 CNNs committee); both models used data augmentation or jittering and had more trainable parameters than this model that only had fewer parameters and one ConvNet. The model performs at an accuracy of 98.87% in the 13th epoch with three spatial transformer layers and SGD without momentum loss optimizer algorithm on BTSC placing second behind GDBM (Yu et al., 2016) and beat OneCNN (Jurišić et al., 2015) and INNLP + SRC (Mathias et al., 2013).

2.3 Traffic Sign Recognition with Traditional Machine Learning

Maldonado-Bascón et al. (2007) used SVM with Gaussian kernels to recognize traffic signs from blobs which have been categorized into shape classes. To test the effect of occlusion on recognition, an occlusion mask was placed on the images. Small, medium-sized and large masks reported 93.24%, 67.85% and 44.90% probabilities of successfully recognizing the signs respectively. An observation made was that a large-sized occlusion mask placed in the middle of the pictogram showed the worst performance during recognition.

Keller et al. (2008) performed real-time traffic sign recognition on U.S. speed limit signs. they assume a unimodal Gaussian distribution for each class and used linear discriminant analysis (LDA) for feature transformation. After that, they used a normal distribution classifier for classification. Their system achieved a combined accuracy of 96.25%; scoring a 98.75% accuracy during detection and 97.5% during classification. This research adopts a CNN to detect 47 classes of U.S. signs of signs that have been already captured in an image.

Larsson et al. (2011) demonstrated the use of Fourier descriptors (FDs) for road sign recognition by using synthetic images of Swedish road signs to create models that were matched against real images using their proposed correlation-based matching method. Local regions were extracted from the synthetic images using the Maximally Stable Extremal Regions (MSER) algorithm proposed by Matas et al. (2004), after which contour sampling of the FDs in the model is matched with the FDs in the extracted query images to find a match. A matching cost based on an empirically set threshold is set and used to accept the match if the cost is less than the threshold. The average precision of the method was reported to be about 95% with few false positives.

Larsson & Felsberg (2011) proposed a method for traffic sign recognition using locally segmented contours described by Fourier descriptors to match prototypes of different traffic sign classes to a query image. A correlation-based matching scheme for Fourier descriptors is used with a fast-cascaded matching scheme for enforcing spatial requirements. The method first extracts Fourier descriptors, it then matches them and matches previously acquired prototypes with spatial models. They tested their method on a dataset of 216 traffic signs collected from 20,000 frames after driving 350km through Sweden. Their method outperformed their earlier results reported in Larsson et al. (2011) for “No Standing or Parking” sign type, with 100% recall and 0 False positives. The recall for the other sign types namely: “Pedestrian Crossing”, “Designated Lane

Right", "50 kph", "30kph", "Priority Road" and "Give Way" were the same as reported in their earlier work Larsson et al. (2011). The false-positive rates were reduced to "0" in the proposed method except for "50 kph" and "30kph" speed signs showing no change from their previous work.

Mathias et al. (2013) recorded an accuracy of 98.53% by combining feature extraction, dimensionality reduction and classification approaches for defining a classification algorithm. Grey-scale values of traffic signs were computed and used with a precomputed pyramid of HOG. This was followed by precomputed HOG1, HOG2, HOG3 features, all making up the feature extraction stage. The resulting dimensions were, for greyscale features – 784-dimensional, the pyramid of HOG – 2172-dimensional descriptors; HOG1 & HOG2 –1568-dimensional and HOG3 – 2916-dimensional. Linear Discriminant Analysis (LDA), Sparse Representation based Linear Projection (SRLP) and Iterative Nearest Neighbors Linear Projection (INNLP) were then employed to reduce the dimensionality of the model. For classification, Nearest Neighbor Classifier (NN), Sparse Representation-based Classifier (SRC), Iterative Nearest Neighbors (INNC) and Support Vector Machines (SVM) were tested. Greyscale + pyramid of HOG + HOGs for feature extraction, INNLP for dimensionality reduction and INNC (K = 62) gave the best Accuracy of 98.53% a slight difference of about 1% less than the GTSRB competition best accuracy. The second-best accuracy of 98.27% was recorded for the Greyscale + pyramid of HOG + HOGs for feature extraction, INNLP for dimensionality reduction and INNC (K = 14). HOG2 with LDA and NN yielded the worst accuracy of 96.97% but had the second-best testing time of 5s following HOG2 with SRLP and LSVM which had a testing performance time of 1s.

Møgelmose et al. (2015) used Integral Channel Features (ICF or ChnFtrs) to detect U.S. traffic signs. The computed features are fed to an AdaBoost classifier with depth-2 decision trees as weak learners. The classifier is then run on the input image using a sliding window. Aggregate

Channel Features (ACF), an improved version of ICF which is faster and has better performance in some instances is also used to classify traffic signs. Both detectors were evaluated on the GTSDB and LISA-TS datasets. The GTSDB dataset is divided into 4 superclasses: “mandatory”, “prohibitory”, “danger” and “other”. ACF performed perfect on danger signs and near perfect on prohibitory signs and mandatory signs; other was ignored. Møgelmose et al. fell short on prohibitory signs, where Mathias et al. scored perfect, scoring an Area Under Curve (AUC) of 99.58/99.86 for ICF/ACF while Mathias et al. scored 100. For mandatory signs, Møgelmose et al. scored 98 .52/98.38 against their 96.98. U.S. signs were divided into four superclasses, namely: “Diamond”, “Stop”, “NoTurn” and “SpeedLimit”. ACF scored 98.98 on “Diamond” and scored above 95 on “NoTurn” and “Stop” whiles “SpeedLimit” scored below 90.

Berkaya et al. (2016) proposed an SVM for classifying traffic signs using the GTSRB. They combined local binary pattern (LBP), HOG and GABOR for feature extraction. For LBP alone, the performance was 93.36%, GABOR alone recorded a performance of 93.90% and HOG alone recorded 94.56%. A combination of all three yielded the best performance of 97.04%. HOG and GABOR together had a performance of 97.00%; close to the performance of all three combined. Their proposed algorithm was ninth overall compared to the results obtained in the 2011 GTSRB competition; however, it had the best performance for “Other Prohibitions”, and “Mandatory”, categories scoring 99.86%, and 99.83% respectively. EBLearn 2LConvNet and CNN HOG3 were the previous best performers in those categories scoring 99.80%, and 97.89 respectively.

Ellahyani et al. (2016) used HOG with HSI, local self-similarity (LSI) together for feature extraction and random forests and SVM as classifiers. They tested their method on GTSDB and Swedish Traffic Sign (STS) datasets and obtained near-state-of-the-art results. Cireşan et al. (2011)

and Sermanet & LeCun (2011) beat their algorithm with 99.46% and 98.31% accuracies respectively, while they registered 97.43% accuracy on the GTSDB dataset. HIS-LSI+HOG with Random Forest yielded the best result. STS dataset had the best recall, precision, and AUC of 93.27%, 90.27%, and 94.05% over GTSDB's 91.07%, 90.13%, and 93.69% respectively.

Soilán et al. (2016) used HOG and SVM to detect and classify 3D traffic signs collected in Spain. The collected signs were divided into seven superclasses: for traffic sign detection namely, omitting direction and information signs. Pedestrian crossing and No Parking", the two most occurring class-specific traffic signs were further classified using a linear SVM model.

Huang et al. (2017) introduced a method for traffic sign recognition (TSR) that extracts HOG features and then feeds them to a single-hidden-layer feedforward network (SLFN) classifier trained on an extreme learning machine (ELM). The algorithm optimizes and generalizes multiclass TSR and can balance the accuracy and computational cost of the model. Different HOG descriptors were evaluated. Competing classifiers were also evaluated with SVM and SVM kernels (with Gaussian kernels) and LDA. HLSGD beats their method when tested on GTSRB with an accuracy of 99.65% against 99.56%; both perform better than the committee of CNNs at an accuracy of 99.46%. The training time for HLSGD, however, is greater than 7 hours while their method takes 209 seconds to train. Kernel ELM has the best performance in the Speed limits, Other prohibitions, Mandatory and Unique categories with 99.54%, 100%; tying Hierarchical SVM with 99.94%, 99.95% respectively. The committee of CNNs recorded the best performance in the Derestriction category with 99.72% over Kernel ELM's 98.33%. On the BTSC dataset, Kernel ELM based scored a recognition accuracy of 98.64% over INNC+INNLP's 98.32%. Kernel ELM achieved an accuracy of 98.12% on rMASTIF dataset beating Kernel SVM, ELM, SVM and LDA-

based methods. Each of these high performing accuracies was obtained using the HOGv+r feature as a descriptor.

Aziz et al. (2018) developed a traffic sign recognition system that extracts HOG, Gabor and compound local binary pattern (CLBP) features from images and feeds them into an ELM. ELM was first introduced by Huang et al. (2006, 2017) as a new learning algorithm for single-layer feedforward neural networks (SFNNs). ELM performs faster than traditional SFNNs like backpropagation because it has fewer parameter tuning and optimum generalization. They evaluated their approach using GTSRB and BTSC and recorded a combined accuracy of 99.10% 98.30% respectively for both datasets. The technique performs at an accuracy of 99.10% better than SVM and K-Nearest neighbor (KNN) with 98.20% & 97.45% respectively on GTSRB. It also performs at an accuracy of 98.30% on BTSC over 97.15% & 96.22% recorded by SVM and KNN respectively.

CHAPTER 3

Methodology

3.1 Approach

Convolutional Neural Network (CNN) has made a lot of stride in the image recognition space in recent times. It gained particular recognition in the traffic sign recognition area during "The German Traffic Sign Recognition Benchmark" (GTSRB) competition, where the best methods presented used CNN for classification (Stallkamp et al., 2011 & 2012).

There is limited research on traffic sign recognition using US traffic signs. malet al. (2016) used an R-CNN algorithm to detect US traffic signs and showed some good results on the LISA-TS Extension dataset. The classification was based on the speed limit superclass alone. This research extends the boundaries of the previous research that adopted the LISA dataset to include all the available classes. It introduces the CIB dataset and trains both datasets on a modified VGGNet. It then compares the results obtained using VGGNet with that of a newly developed deep learning model, NuNet, that trains faster and performs better.

The experimental setup consists of two main parts. Firstly, preparing the data, that spans all the tools used to extract and separate traffic signs into their respective classes. Secondly, the extracted data is fed to a deep neural network for classification. Experiments were carried out in Python 3 environment running in Jupyter Notebook (Kluyver et. al., 2016) hosted on a 64-bit desktop computer with 64 cores of 4GB RAM each. Linux 18.04 is the operating system on which all experiments were conducted. TensorFlow (Abadi et. al., 2016) is an open-source machine learning framework for dataflow representations that have been widely adopted for machine learning research. TensorFlow has been used to develop several industry applications used by companies such as Google, Airbnb, eBay, and Intel, and was adopted in this research.

3.2 Data Preparation and Exploration.

The LISA TS dataset is a publicly available U.S. traffic sign dataset (Møgelmose et al., 2012) that can be used to conduct research involving U.S. traffic signs to test the performance of models. The LISA-TS extension dataset provides additional data points for traffic sign recognition and was adopted for this research. This study also introduces a new dataset, the CIB TS-V1 (Center of Identity and Biometrics Traffic Sign Version 1) for comparison of the performance of the two models.

3.2.1 LISA Dataset

Traffic sign datasets for several countries have been published and made publicly available for researchers to train and test their models against published models. Álvaro Arcos-García et al. (2017) introduced a Spanish dataset, the Belgium dataset was reported (Timofte et al., 2014), Germany (Stallkamp et al., 2011), Croatia (Jurišić et al., 2015), Italy (Youssef et al., 2016), Sweden (Larsson & Felsberg, 2011), and China (Zhu et al., 2016).

The zipped LISA TS dataset folder comes with a set of Python tools for extracting traffic sign annotations from full frames. The “categories.txt” is a comma delimited text file (see Table 3.1) containing superclasses of traffic signs, namely: “warning”; “prohibition”; “speedLimit” and; “speedLimitGood”. The “extractAnnotations.py” is a Python file that provides methods for copying, marking, blackout and cropping regions of interest (ROI). After extracting annotations, the “mergeAnnotationFiles.py” file is then used to merge multiple annotation files into one file. The “evaluateAnnotation.py” is a Python file for detecting traffic signs – beyond the scope of this work. The cropping feature of the “extractAnnotations.py” file was used to crop traffic signs for classification for this work.

Table 3.1. Super Class and Corresponding Classes

warning	**prohibition**	**speedLimit**	**speedLimitGood**
addedLane	doNotPass	speedLimit15	speedLimit15
curveRight	keepRight	speedLimit25	speedLimit25
dip	rightLaneMustTurn	speedLimit30	speedLimit30
intersection	speedLimit15	speedLimit35	speedLimit35
laneEnds	speedLimit25	speedLimit40	speedLimit40
merge	speedLimit30	speedLimit45	speedLimit45
pedestrianCrossing	speedLimit35	speedLimit50	speedLimit50
signalAhead	speedLimit40	speedLimit55	speedLimit55
slow	speedLimit45	speedLimit65	speedLimit65
stopAhead	speedLimit50	speedLimitUrdbl	
thruMergeLeft	speedLimit55		
thruMergeRight	speedLimit65		
turnLeft	truckSpeedLimit55		
turnRight			
yieldAhead			

LISA TS benchmark is constituted of 7,855 traffic sign images categorized into 47 traffic sign classes as listed below:

“addedLane”, “curveRight”, “doNotEnter”, “doNotPass”, “intersection”, “keepRight”, “laneEnds”, “merge”, “noLeftTurn”, “school”, “noRightTurn”, “pedestrianCrossing”, “rampSpeedAdvisory20”, “dip”, “rampSpeedAdvisory35”, “rampSpeedAdvisory40”,

“rampSpeedAdvisory45”, “rampSpeedAdvisory50”, “curveLeft”, “rampSpeedAdvisoryUrdbl”, “rightLaneMustTurn”, “roundabout”, “schoolSpeedLimit25”, “signalAhead”, “slow”, “speedLimit15”, “speedLimit25”, “speedLimit30”, “speedLimit35”, “speedLimit40”, “speedLimit45”, “speedLimit50”, “speedLimit55”, “speedLimit65”, “speedLimitUrdbl”, “stop”, “stopAhead”, “thruMergeLeft”, “thruMergeRight”, “turnRight”, “thruTrafficMergeLeft”, “truckSpeedLimit55”, “turnLeft”, “yield”, “yieldAhead”, “zoneAhead25”, “zoneAhead45”.

The LISA TS extension is made up of 2,498 traffic sign images comprising 19 traffic sign classes as listed below:

“Stop”, “doNotEnter”, “pedestrianCrossing”, “speedLimit30”, “noParking”, “speedBumpsAhead”, “speedLimit15”, “speedLimit25”, “curveRight”, “signalAhead”, “speedLimit35”, “stopAhead”, “yieldToPedestrian”, “bicyclesMayUseFullLane”, “noLeftAndUTurn”, “curveLeft”, “intersectionLaneControl”

The two datasets were combined into a single dataset resulting in 10,353 traffic sign images categorized into 53 classes as listed below:

“addedLane”, “bicyclesMayUseFullLane”, “curveLeft”, “curveRight”, “dip”, “doNotEnter”, “doNotPass”, “intersection”, “intersectionLaneControl”, “keepRight”, “laneEnds”, “merge”, “noLeftAndUTurn”, “noLeftTurn”, “noParking”, “noRightTurn”, “pedestrianCrossing”, roundabout”, “rampSpeedAdvisory20”, “rampSpeedAdvisory35”, “rampSpeedAdvisory40”, “rampSpeedAdvisory45”, “rampSpeedAdvisory50”, “rampSpeedAdvisoryUrdbl”, “school”, “rightLaneMustTurn”, “ “schoolSpeedLimit25”, “signalAhead”, “slow”, “speedBumpsAhead”, “speedLimit15”, “speedLimit25”, “speedLimit30”, “speedLimit35”, “speedLimit40”, “speedLimit45”, “speedLimit50”, “speedLimit55”, “speedLimit65”, “speedLimitUrdbl”, “stop”, “stopAhead”, “thruMergeLeft”, “thruMergeRight”, “thruTrafficMergeLeft”, “turnLeft”,

“turnRight”, “truckSpeedLimit55”, “yield”, “yieldAhead”, “yieldToPedestrian”, “zoneAhead25”, “zoneAhead45”.

This list was further cleaned to exclude unreadable speed signs, “speedLimitUrdbl” & “rampSpeedAdvisoryUrdbl” and classes with less than 10 samples. 8,855 traffic sign annotations resulted belonging to the 38 classes is listed below:

“addedLane ,”curveLeft “, “curveRight “, “dip “, “doNotEnter “, “keepRight “, “laneEnds “, “merge “, “noLeftTurn “, “noRightTurn “, “pedestrianCrossing “, “rampSpeedAdvisory20 “, “rampSpeedAdvisory45 “, “rampSpeedAdvisory50 “, “rightLaneMustTurn “, “roundabout “, “school “, “schoolSpeedLimit25 “, “signalAhead “, “slow “, “speedLimit15 “, “speedLimit25 “, “speedLimit30 “, “speedLimit35 “, “speedLimit40 “, “speedLimit45 “, “speedLimit50 “, “speedLimit65 “, “stop “, “stopAhead “, “thruMergeLeft “, “truckSpeedLimit55 “, “turnLeft “, “turnRight “, “yield “, “yieldAhead “, “zoneAhead25 “, “zoneAhead45”

First, the dataset is merged into one CSV file named “mergedAnnotations.csv” using the “mergeAnnotationFiles.py” file by running the command, “python mergeAnnotationFiles.py frame mergedAnnotations.csv annotations/”. This command combines annotation tiles in any subdirectory matching a regex pattern of the image filenames. The merged files are copied into a file called “allAnnotations.csv” using the commands: "python extractAnnotations.py -c [category] copy allAnnotations.csv". the categories are “warning”, “prohibition”, "speedLimits” and “speedLimitGood”.

The merged annotations are then split into 2 CSV files using the “splitAnnotaionsFiles.py” file into “split1.csv” and “split2.csv”. A split percentage of 80% was specified for “split1.csv” as the training set and the remaining annotations were put in “split2.csv” for validation. Annotations in the two files, “split1.csv” and “split2.csv”, were then extracted into their respective class

subfolders under the training and validation folders, respectively. After the split, the "extractAnnotations.py" Python file was used to crop and copy image annotations in each split into their respective subfolders using the class label as the name of the subfolder. This exercise was performed on both splits. The merging of LISA TS and LISA TS extension datasets resulted in 38 classes that were cropped and copied into their respective class folders using the naming convention: "addedLane", "curveLeft", "curveRight", …, "yieldAhead", "zoneAhead25", "zoneAhead45". These subfolders were organized in the folders called "training" and "validation", representing training and validation sets.

Figure 3.1 shows the distribution of traffic signs for each class in the entire dataset while figures 3.2 and 3.3 show the distribution of signs in the training and validation sets, respectively. The most populated classes are "stop", "pedestrianCrossing", and "signalAhead" in an order of decreasing magnitude. The least occurring traffic signs in the dataset are "rampSpeedAdvisory20", "curveRight", "addedLane", and "thruMergeRight".

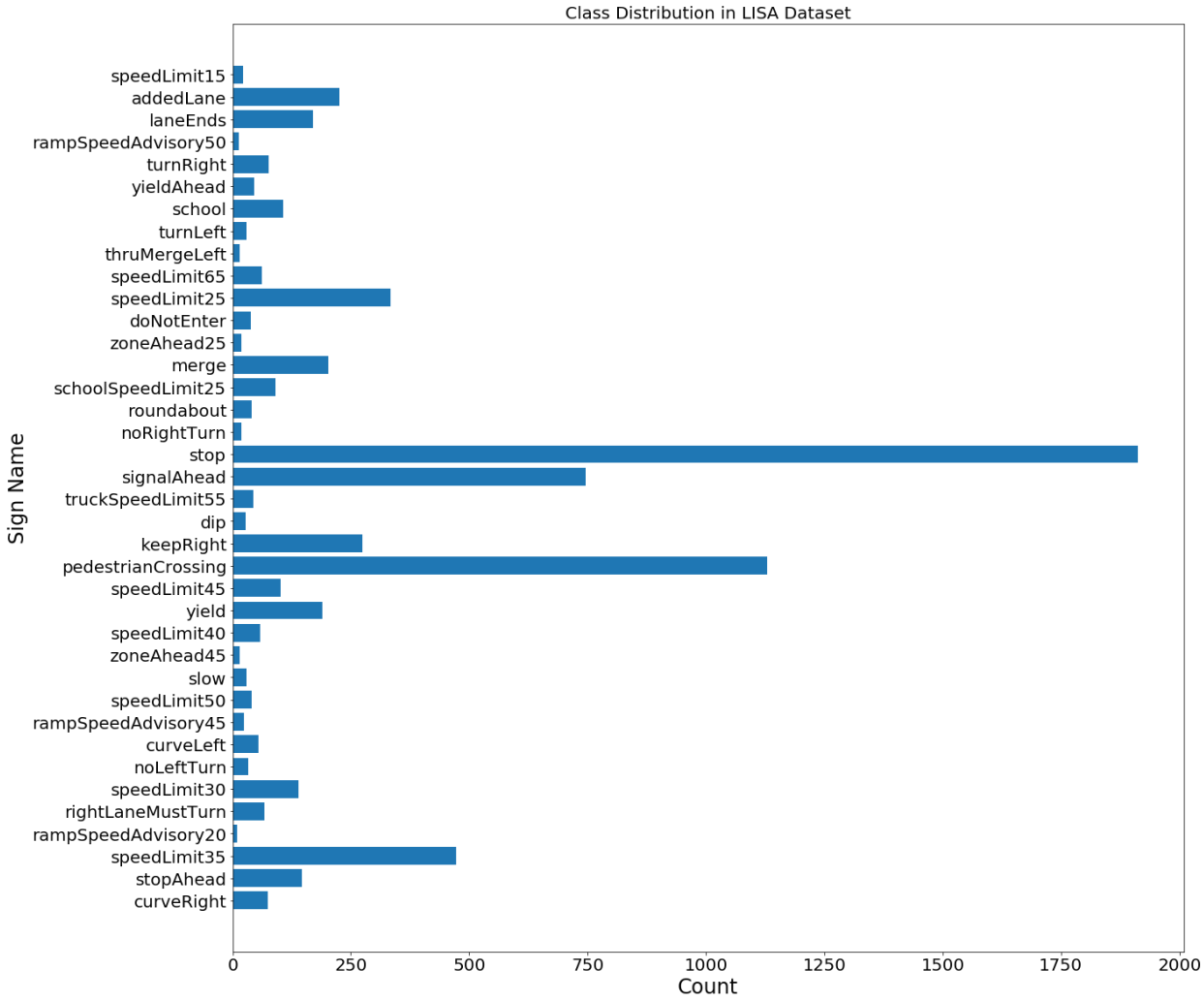

Figure 3.1. Traffic Sign Data Distribution per Class in the Entire LISA Dataset

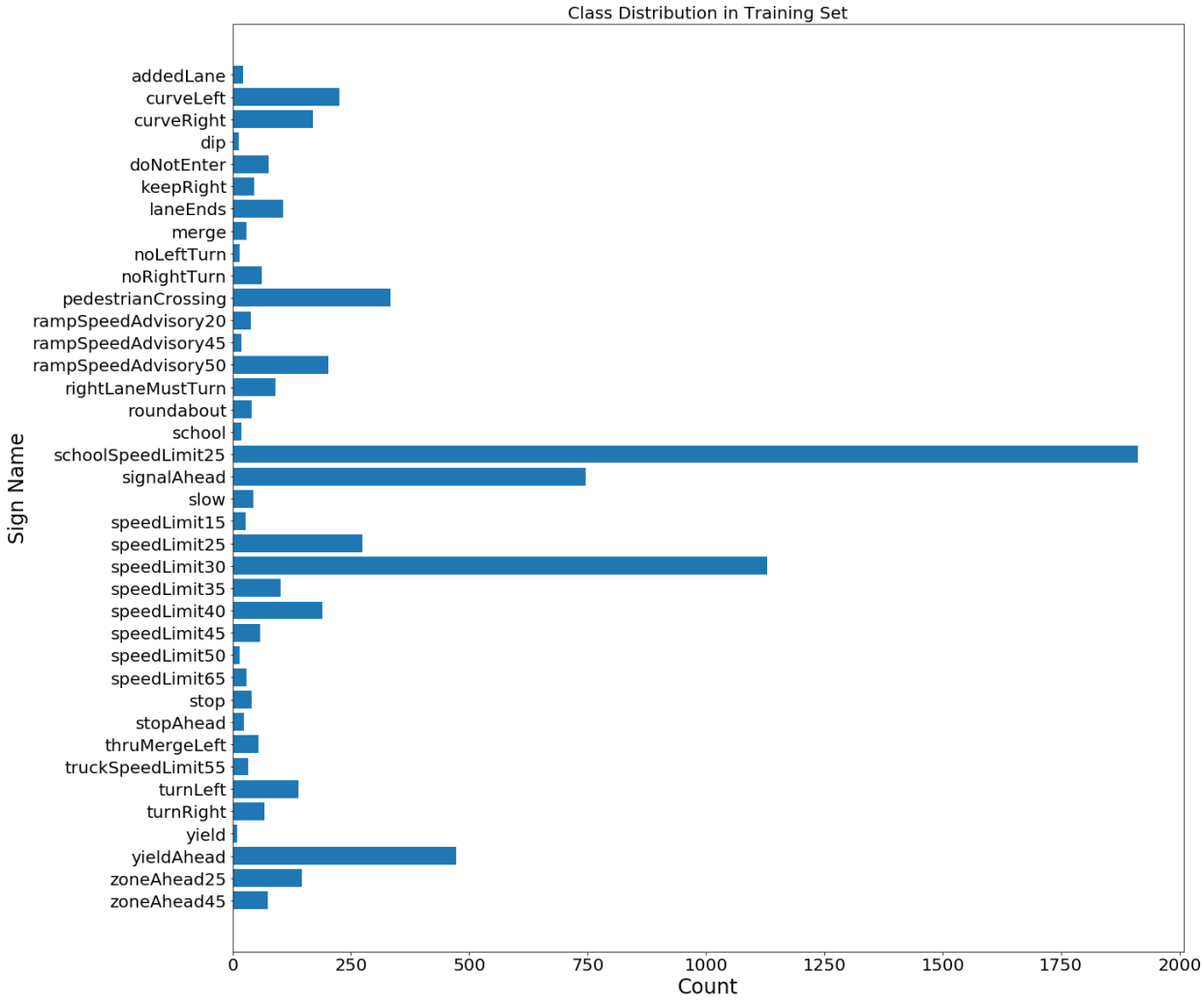

Figure 3.2. Traffic Sign Distribution per Class in Training Set LISA Dataset

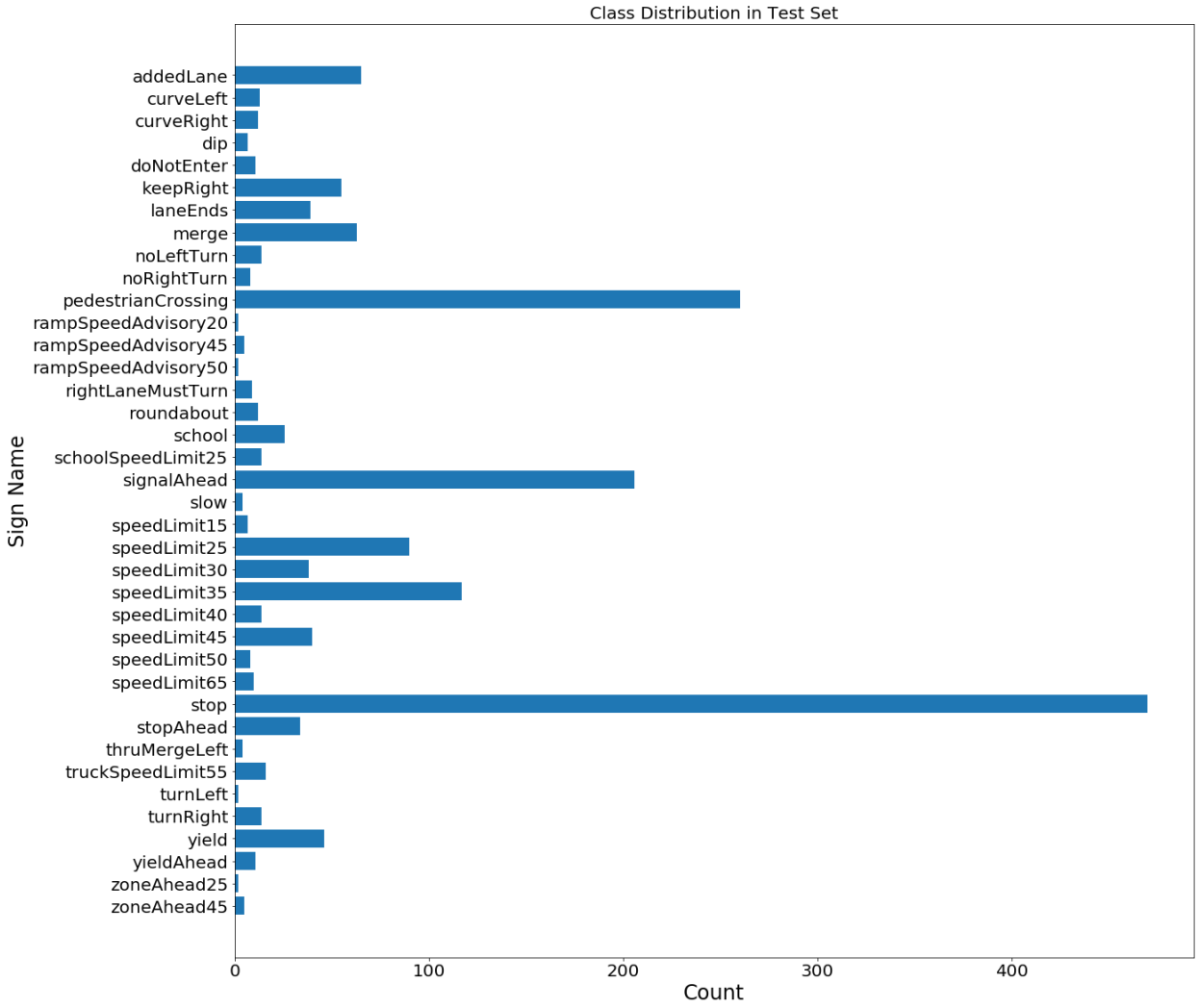

Figure 3.3. Traffic Sign Distribution per Class in validation Set LISA Dataset

Figure 3.4 shows samples of traffic sign images in each class. The images shown are the first images in that class. It should be noted that several of these signs have numbers showing speed limits of some sort. This raises concern for the performance of the model especially with the limited samples in each class.

Figure 3.4. Sample Image in Each Class in LISA Dataset

3.2.2 Cyber Identity Biometrics Traffic Sign (CIB TS) Dataset

An industrial camera was attached to a sedan and driven around sections of the Greensboro city, North Carolina. The dataset consists of 690 traffic signs images representing 9 classes of U.S. traffic signs; this constitutes the first version of the Cyber Identity and Biometrics traffic sign (CIB TS-V1) dataset. CIB is a Computer Science laboratory at North Carolina A&T State University. Traffic sign annotation was done using a publicly available tool called "labelImg" (Tzutalin, 2015). Traffic signs were cropped out of the image frames and saved in their respective classes. Sample images per class are shown in figure 3.5.

A python script was used to split the images into training and validation sets in an 80/20 ratio respectively. The traffic sign classes present in the CIB dataset are “speedLimit20”, “curveRight”, “noTruck”, “speedLimit30”, “pedestrianCrossing”, “bicycleLane”, “height12-9”, “doNotEnter”, and “stop”. The class distribution for the entire CIB dataset is presented in figure 3.6. Figures 3.7 and 3.8 represent the distribution per class for the training and validation sets respectively.

Figure 3.5. Sample Image in Each Class in CIB Dataset

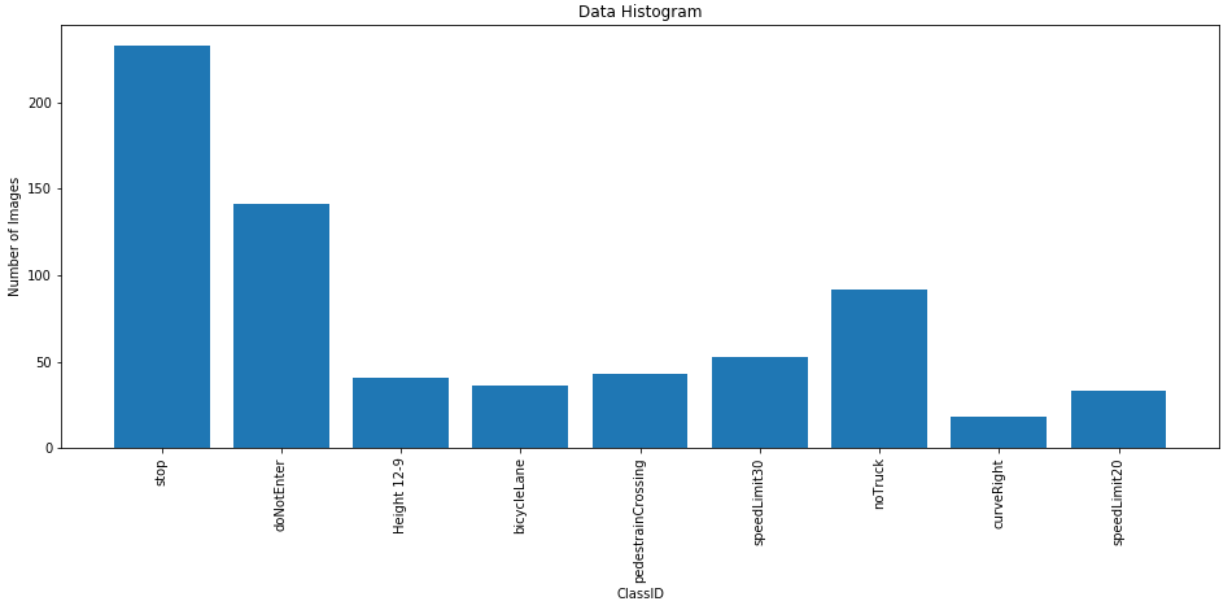

Figure 3.6. Traffic Sign Distribution per Class of whole CIB Dataset

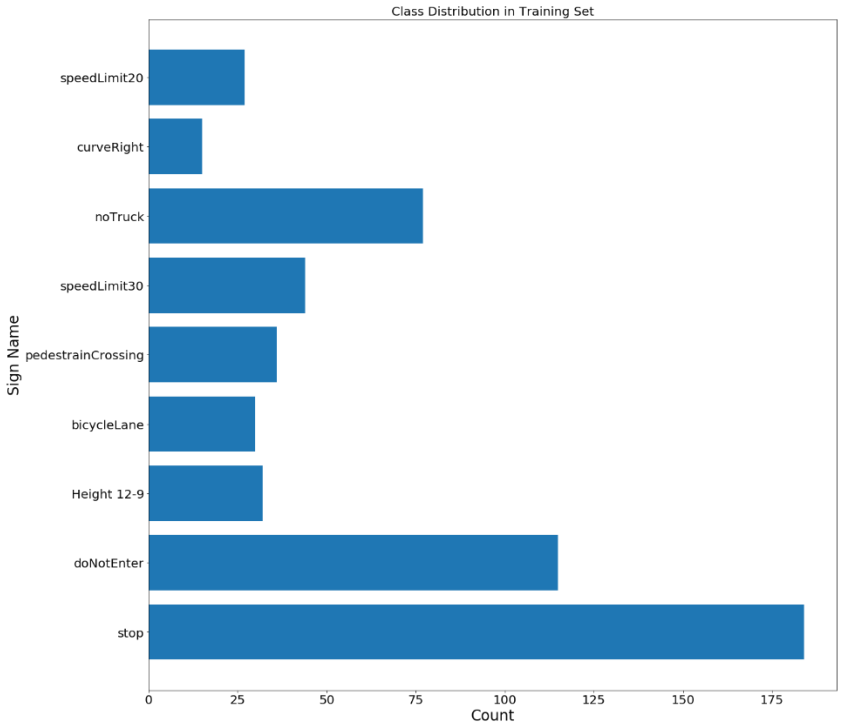

Figure 3.7. Traffic Sign Distribution per Class in Training Set of CIB Dataset

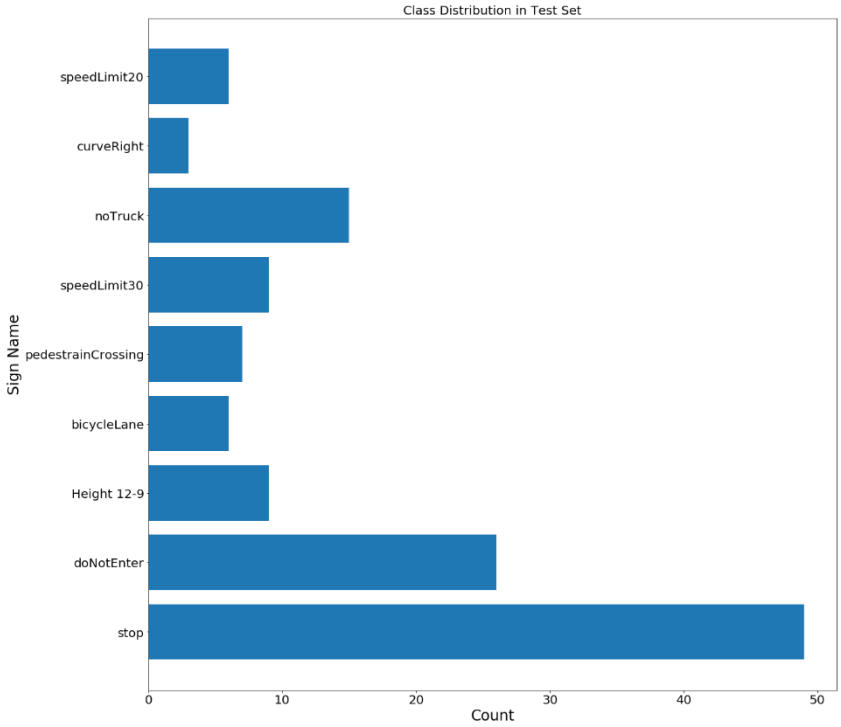

Figure 3.8. Traffic Sign Distribution per Class in validation Set of CIB Dataset

3.3 VGGNet Architecture

The VGG Network adopted from Simonyan & Zisserman (2015) has a batch normalization layer for faster and better training, a param ReLu layer for solving dead linear rectifier issue during training, convolution layer with parametric relay activation using Xavier Scheme for weights and biases initialization, fully connected layer with fully connected dense layers also using Xavier Scheme for weights and biases initializations and a max-pooling operation as its basic elements.

The VGG layer (see Figure 3.9) is made up of 2 back to back Convolutions of 2x2 kernel size and a stride of 2. Each layer has 1 pooling layer followed by a dropout layer. The model is made up of 4 VGG layers. The first layer extracts 32 feature maps from an input layer of 32x32x3 features while the second layer extracts 64 feature maps from a 16x16x32 input, with the third layer extracting 128 feature maps from an 8x8x64 input and the fourth, 256 feature maps from an input of 4x4x128. The model also has 3 fully connected layers with the 1024 hidden layers that map to 512 hidden layers that further map to the number of classes in each dataset.

Model building parameters, including Xavier Initialization as a tuning methodology, early stopping to restore the previous checkpoint if accuracy gains in testing do not meet current requirements, a learning rate of 5e-5, and a regularization factor of 1e-3 are chosen as the best choice for hyperparameters after searching over 600 Epochs. Adam Optimizer was used for optimization. A batch size of 500 was chosen for the LISA dataset and 100 for the CIB dataset. The modified VGGNet was adopted from a publicly available Python code on GitHub[1].

[1] https://github.com/vamsiramakrishnan/TrafficSignRecognition/blob/master/TrafficSignClassifier.ipynb

3.4 VGGNet Feature Extraction

The first stage of the VGGNet has 2 layers and extracts 32 feature maps from an input layer of 32x32x3 features while the second stage has 3 layers and extracts 64 feature maps from a 16x16x32 input. The third stage also has 3 layers extracting 128 feature maps each from an 8x8x64 input and the fourth, 256 feature maps from an input of 4x4x128. The model also has 3 Fully Connected layers with the 1024 hidden layers that map to 512 hidden layers and further map to a softmax layer which the number of classes in each dataset. Features from each layer are presented by figures 3.10-3.12.

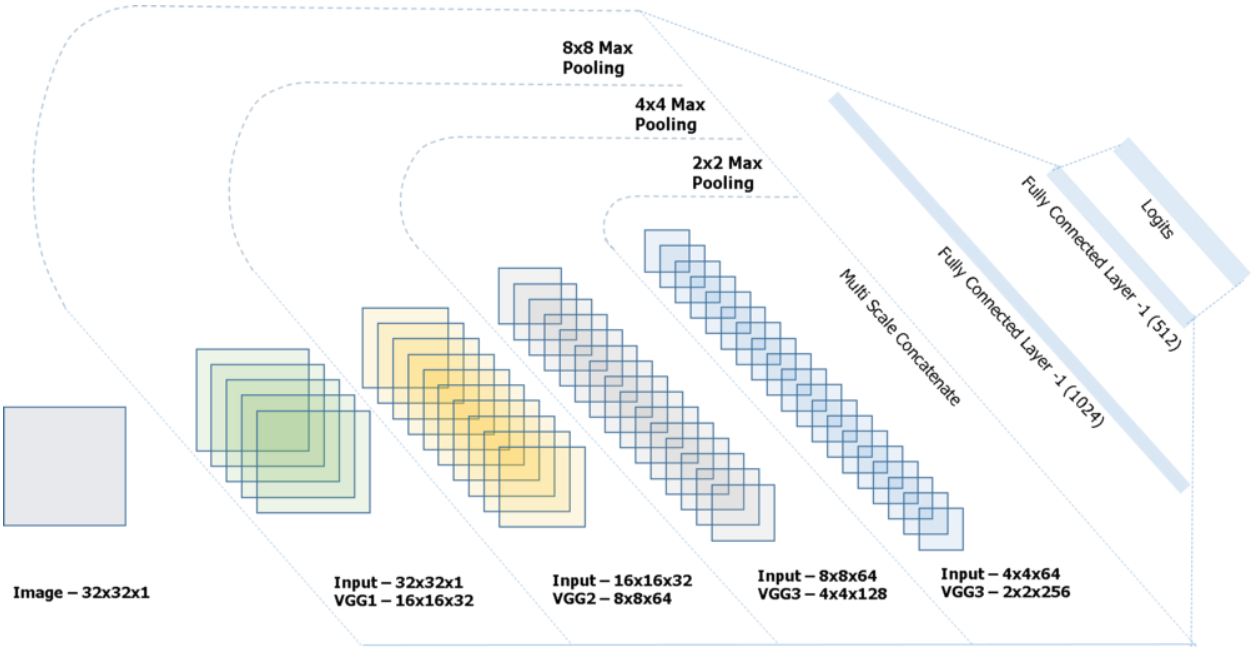

Figure 3.9. VGGNet Architecture[1]

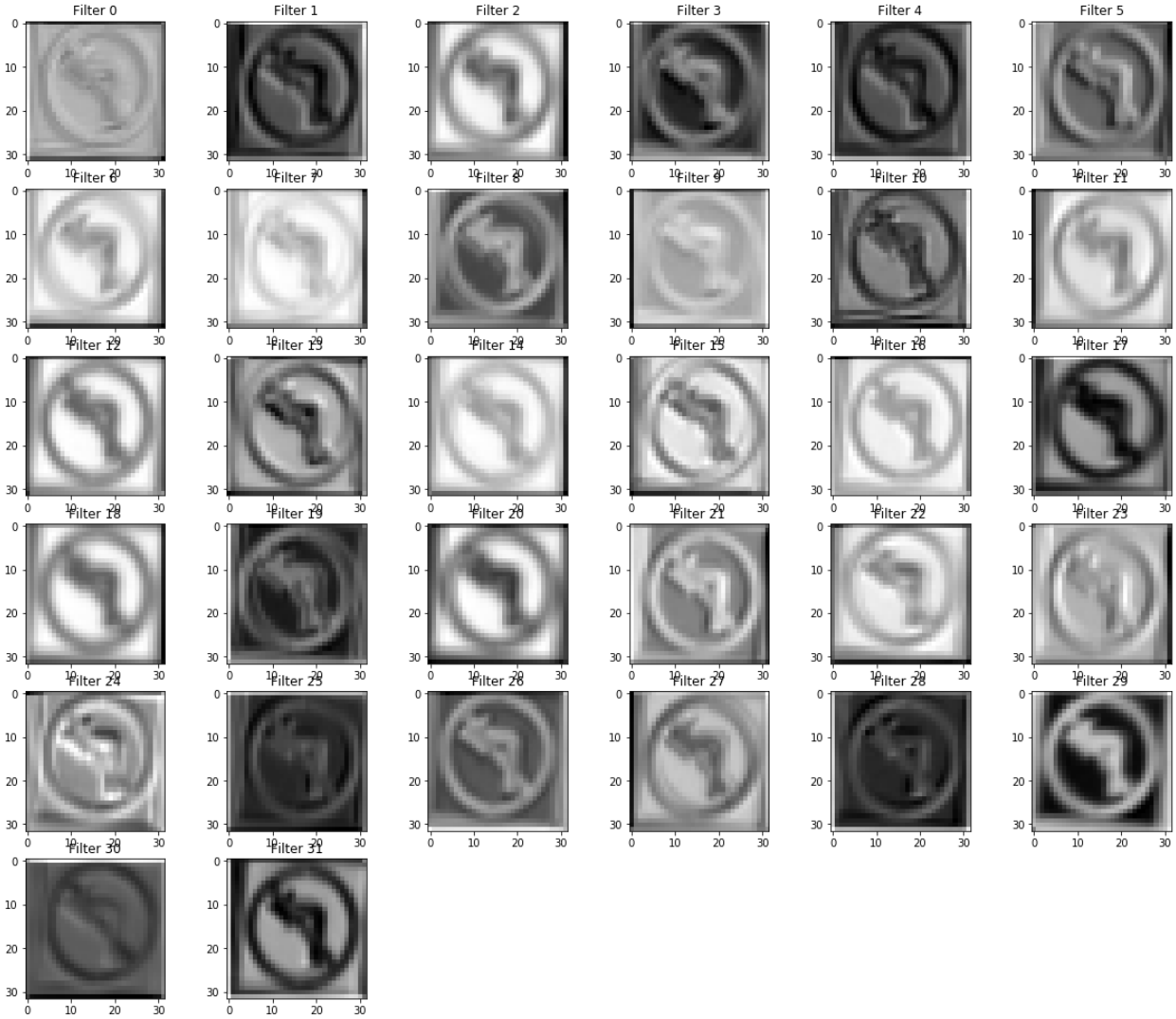

Figure 3.10. Features Extracted from First Layer of the First Stage of VGGNet Model

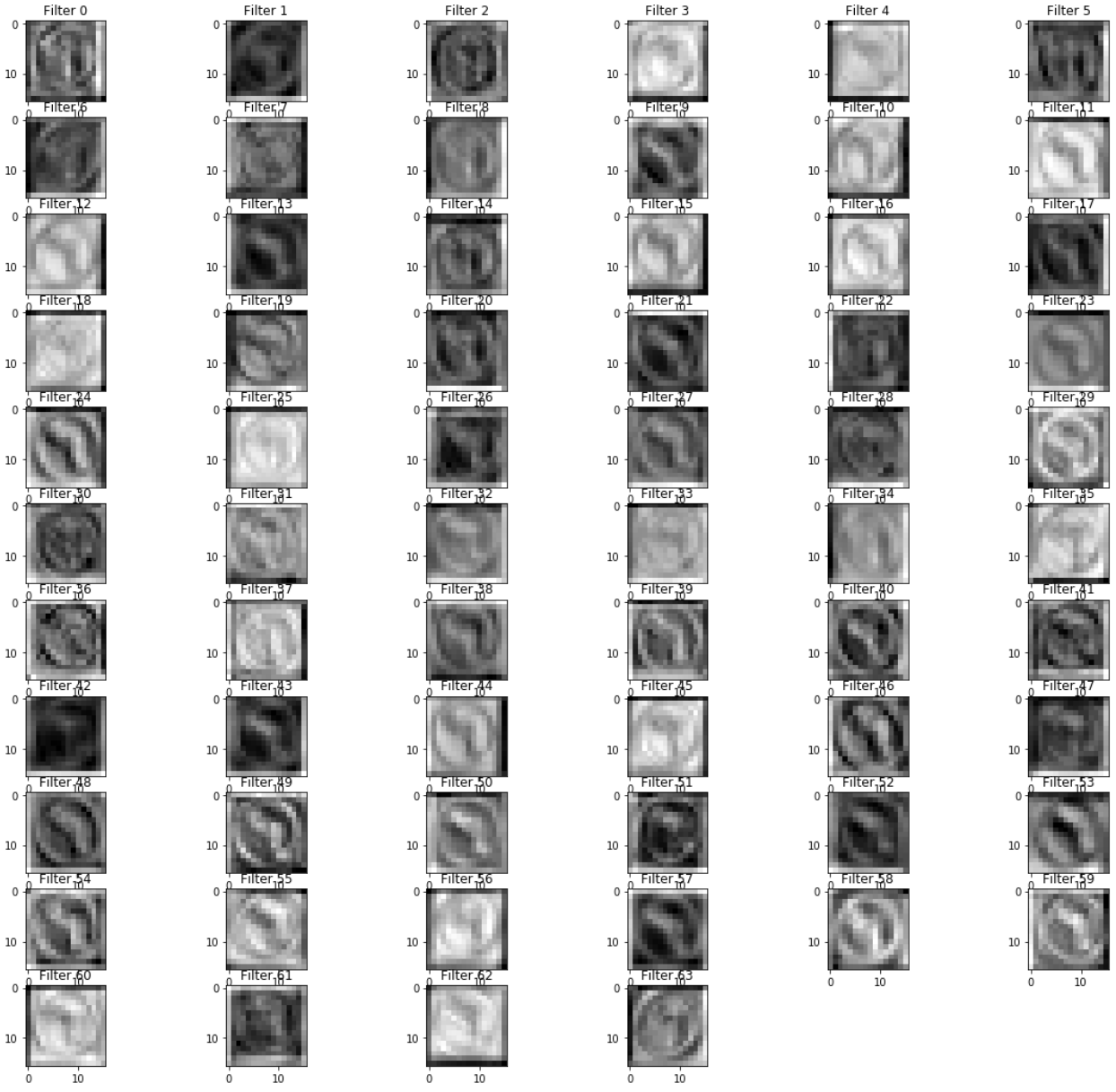

Figure 3.11. Features Extracted from First Layer of the Second Stage of VGGNet Model

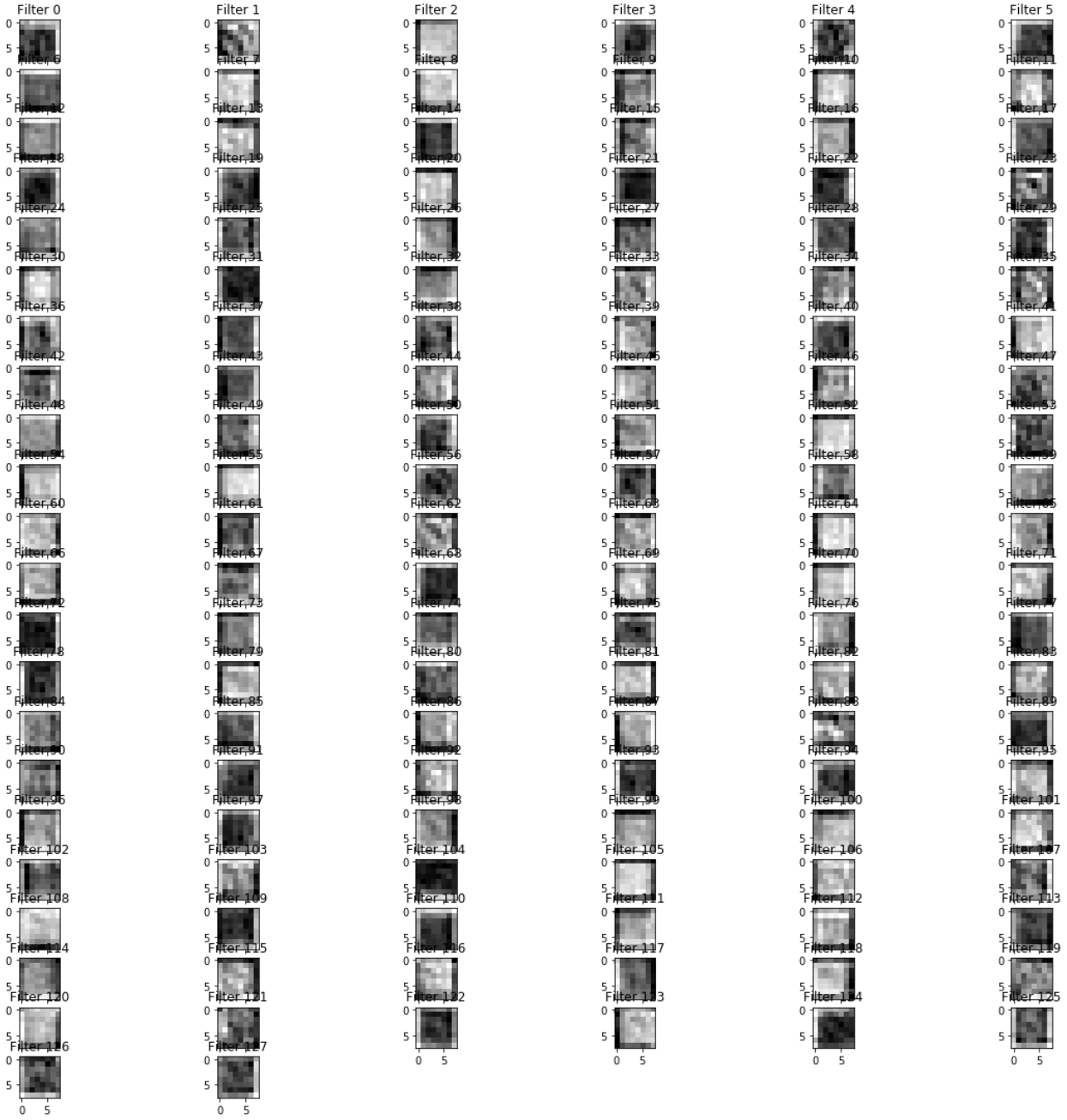

Figure 3.12. Features Extracted from First Layer of the Third Stage of VGGNet Model

3.5 Results of Training VGGNet on LISA Dataset

The results from training VGGNet on LISA are presented in the section. The split ratio composed of 80% of the dataset being used as the training set and 20% as the validation set. Training VGGNet on LISA took approximately 26 hours after which the accuracy and loss plots

were obtained. Figure 3.13 shows a plot of the training and validation accuracies of the model. The loss plots for training and validation are presented in Figure 3.14 with detailed results of the loss and accuracy for each epoch is recorded in a table in appendix A.

The original research recorded an accuracy of 98.7% on the GTSRB dataset. In the current work, the model performs at a training accuracy of 99.93% and a validation accuracy of 99.83% on the LISA dataset. The losses recorded for the training and validation sets were 6.6% and 7.1% respectively. The original VGGNet showed a loss of 7.5% during validation on GTSRB.

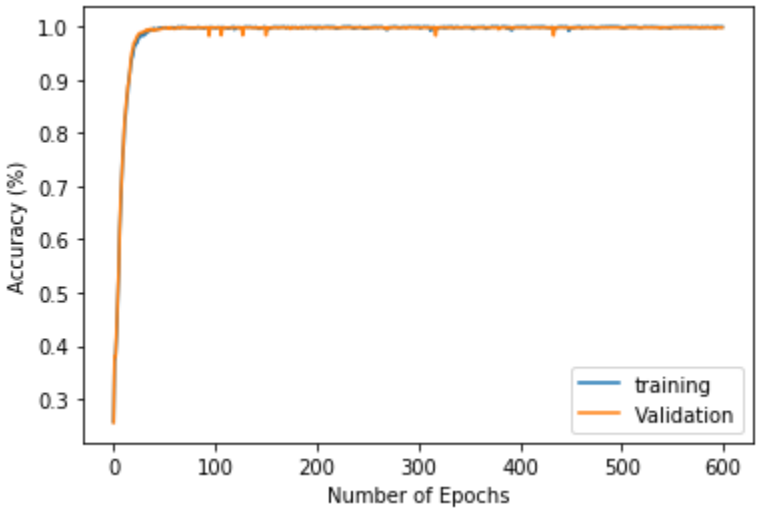

Figure 3.13. A Plot of Training and Validation Accuracies for VGGNet on LISA dataset

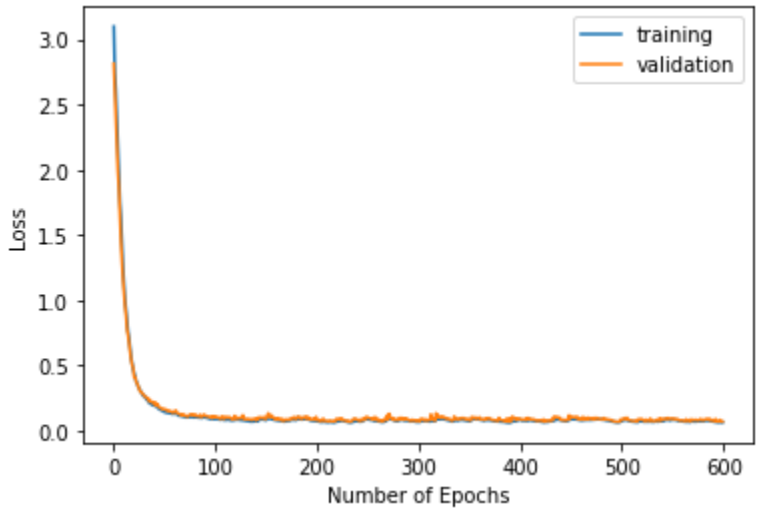

Figure 3.14. A Plot of Training and Validation Losses for VGGNet on LISA dataset

The most occurring class in the dataset is “stop”. It was 100% accurately classified after the first epoch, while most of the other classes were misclassified as “stop”. This important observation follows that the model is able to learn relevant details from numerous samples and apply them to recognize the signs better than fewer samples. The confusion matrix presented in figure 3.15 shows which classes in the dataset the model classified correctly or misclassified. Only 3 signs in the “speedLimit15” were misclassified as “pedestrainCrossing”. All other signs were perfectly classified into their respective classes. Intermediate confusion matrices were plotted and presented in appendix B.

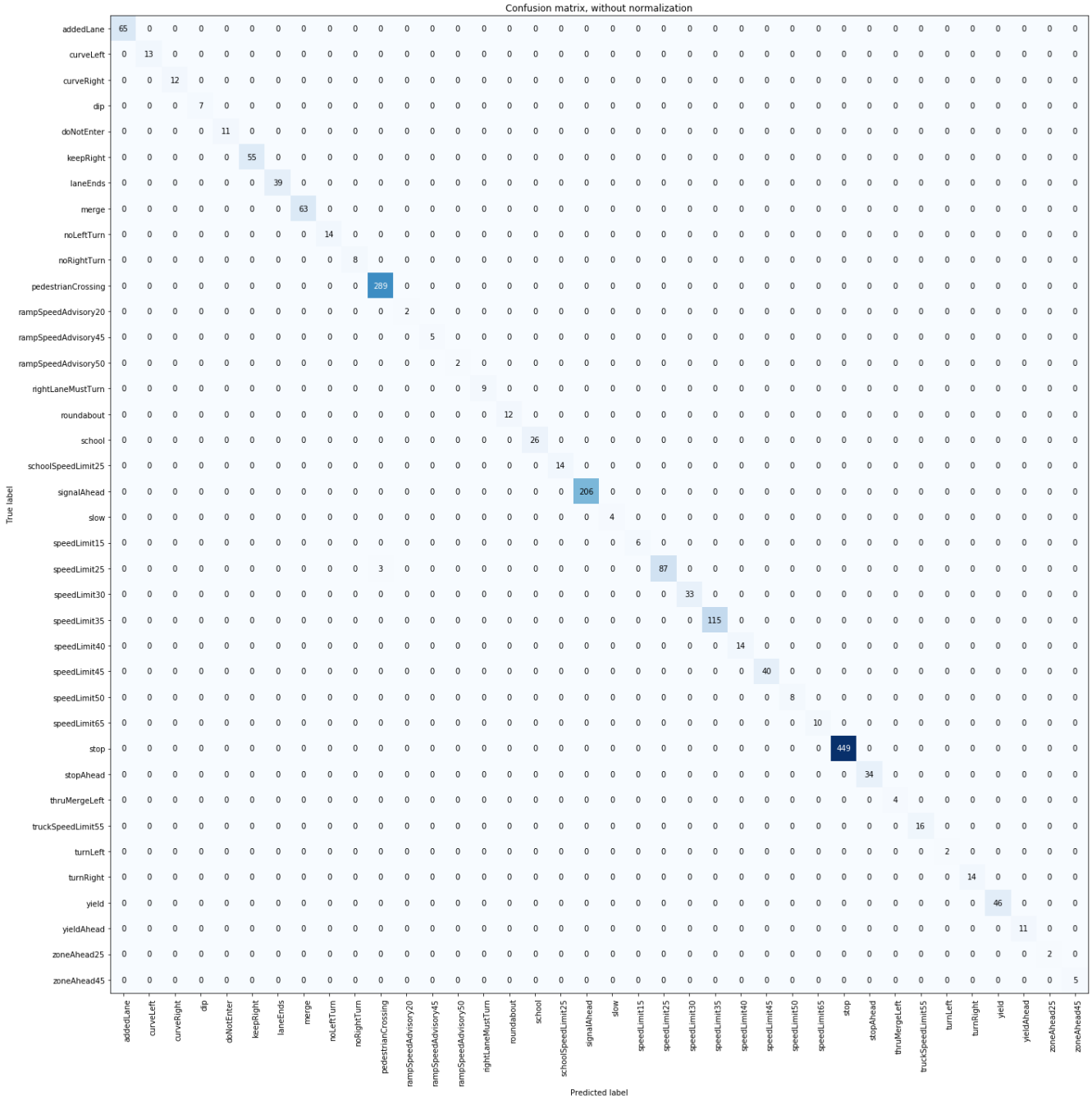

Figure 3.15. Confusion Matrix of VGGNet Model after Training on LISA Dataset

3.6 Result of Training VGGNet on CIB TS V1 Dataset

The result from training VGGNet on the CIB dataset is presented in this section. A training to validation split ratio of 80/20 was used. The model recorded an accuracy of 100% on the training set and records a validation accuracy of 96.92% on the CIB dataset. An error of 1.8% was recorded

for the training samples and a validation error of 5.6% was recorded. It took approximately 9 hours to train the model. VGGNet overfits the CIB dataset as it begins to learn unnecessary details down its depth and cannot generalize well on unseen data in the validation set. A deep model requires a lot of data to be able to generalize classification to unseen data. The depth of VGGNet coupled with the data input size makes it overfit. It can recognize data it has already encountered in the training set but cannot properly recognize those in the validation set. The accuracy and loss plots for the training and validation sets are presented in Figures 3.16 and 3.17, respectively. These plots show how unstable the VGGNet performs on the CIB dataset. It also shows that VGGNet does not reach a global maximum.

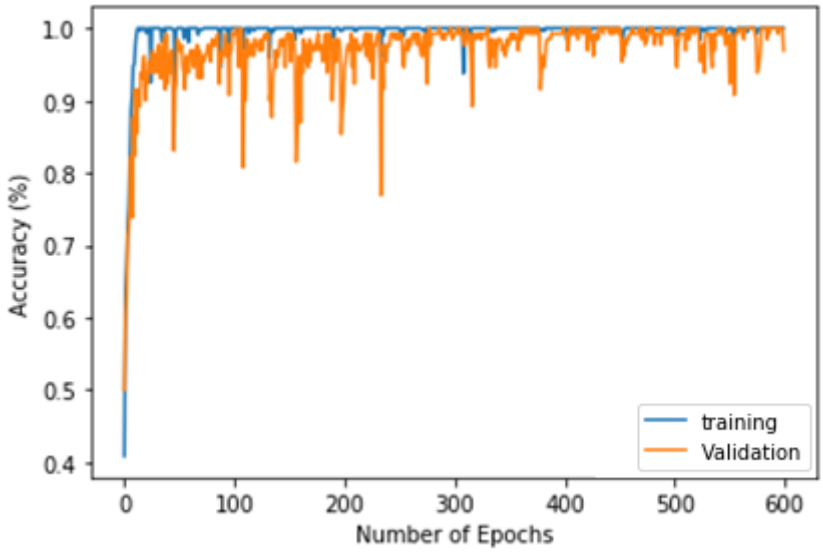

Figure 3.16. A Plot of Training and Validation Accuracies for VGGNet on CIB Dataset

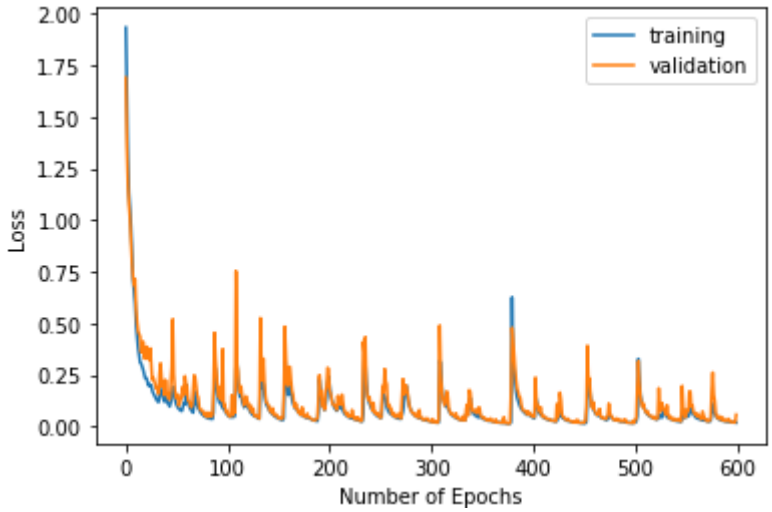

Figure 3.17. A Plot of Training and Validation Losses for VGGNet on CIB Dataset

Since the training was not yielding a reliable accuracy or loss, it became pertinent to design a model that performs well on smaller as well as larger datasets. This is one of the motivations for the development of the NuNet, presented in the next chapter. The confusion matrix of the experiment is presented in Figure 3.18 showing the classes in the validation set which were correctly classified or misclassified. It should be noted that the point maximum at epoch 600 is not the global maximum. Even though the confusion matrix mimics a perfect classification, subsequent epochs may misclassify samples.

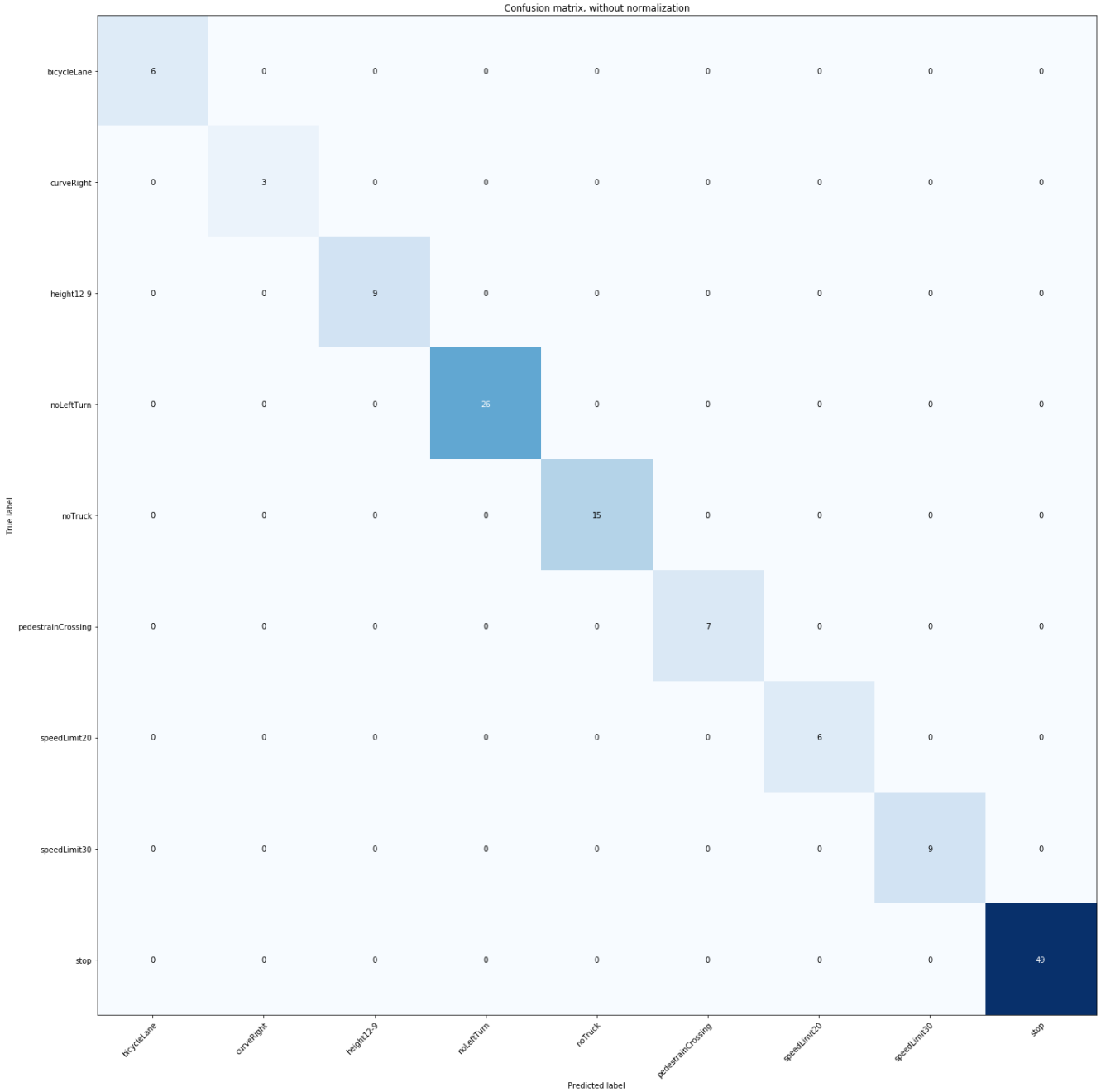

Figure 3.18. Confusion Matrix of Model after Training VGGNet on CIB Dataset

3.7 Discussion

The VGGNet is a deep neural network that extracts robust features from an image. While this architecture offers a comprehensive selection of features, it also poses the danger of extracting unnecessary information; especially when the training data is too small. In the case of a large

training set, the VGGNet is stable and generalizes well. However, when a smaller dataset is presented, the model overfits as it overlearns from the few data it has already seen and is not able to generalize very well to new unseen data. This issue can be addressed by augmenting the data, adding more data samples to the training set or reducing the depth of the model. The first two are not necessarily of interest to this research. This research seeks to develop a robust model that can both perform when there is enough data for training or not, thereby addressing the issue of data size requirement for deep learning object recognition.

There is a tradeoff between model accuracy and training time. Whereas Feeding the model with lots of data improves the performance of the VGGNet model, it follows that it takes a longer time to train the network. With modern improvements in hardware, the latter can be addressed, however, it comes at a high financial cost. The VGGNet took approximately 26 hours to train 8,855 samples over 600 epochs on the LISA dataset. On CIB, VGGNet took approximately 9 hours to train 552 training samples over 1000 epochs.

CHAPTER 4

NuNet Model Architecture and Results

4.1 NuNet Architecture

NuNet is a deep learning approach designed to tackle the issue of poor model performance on a limited dataset and improving speed. The proposed model is a light-net architecture that has the potential to update its parameters during training. The model starts with a single convolution followed by a max-pool operation, then two fully connected layers and then a SoftMax. The model takes a 32x32x3 input image and performs a convolution with a 3x3 filter and a 2x2 stride and outputs 16 features as shown in Figure 4.1. A 2x2 max-pooling operation follows the convolution, whose result is passed through two fully connected layers; 256x1 and 128x1. A SoftMax function is then used to classify the result into the number of classes.

Single-layer convolutions followed by a max-pool operation train and converge faster as fewer computations are required to extract features. At some point during training, improvements in the accuracy and reduction of the error/loss stall. The model then makes changes to parts of its components, such as the convolutional (filter) size, the number of layers and the number of filters (output layers). These modifications then enhance performance during training, thereby improving speed and accuracy.

During training, when an error reduction stalls, the model was configured to increase the number of convolutional layers. Figures 4.2 and 4.3 represent a second and third convolutional layers added before a max-pool operation.

The model was tested on LISA dataset and CIB-V1 with results presented below:

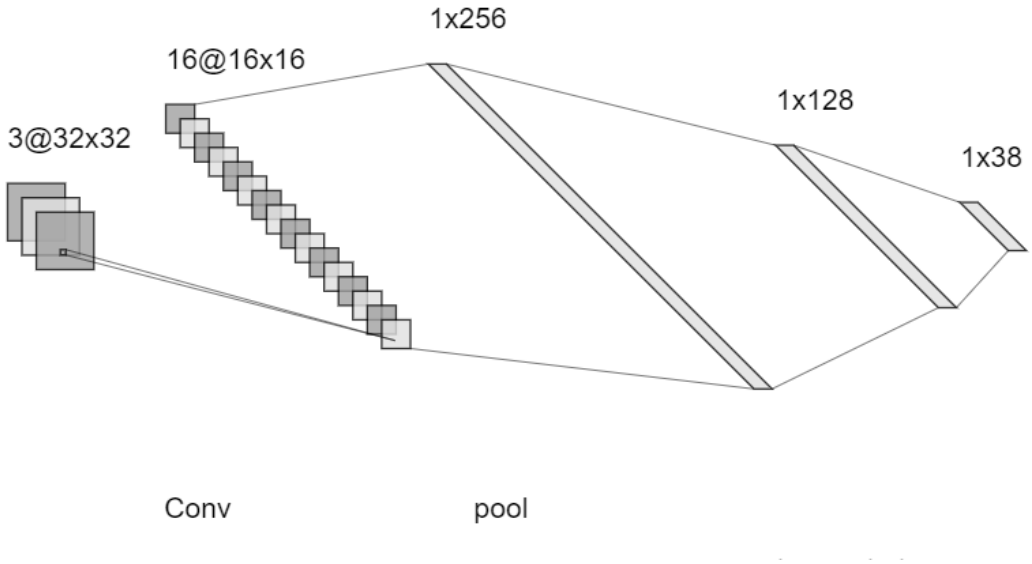

*Figure 4.1.*NuNet Model Architecture with a Single Layer

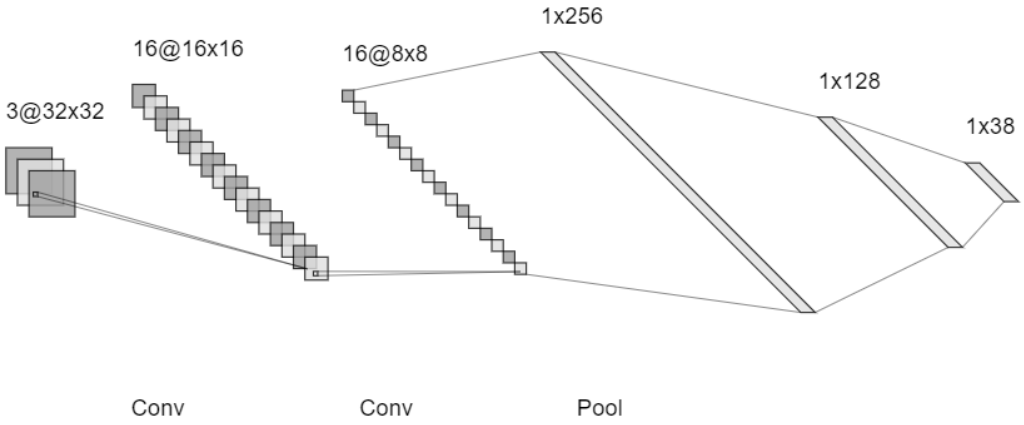

Figure 4.2. NuNet Model Architecture with Two Layers

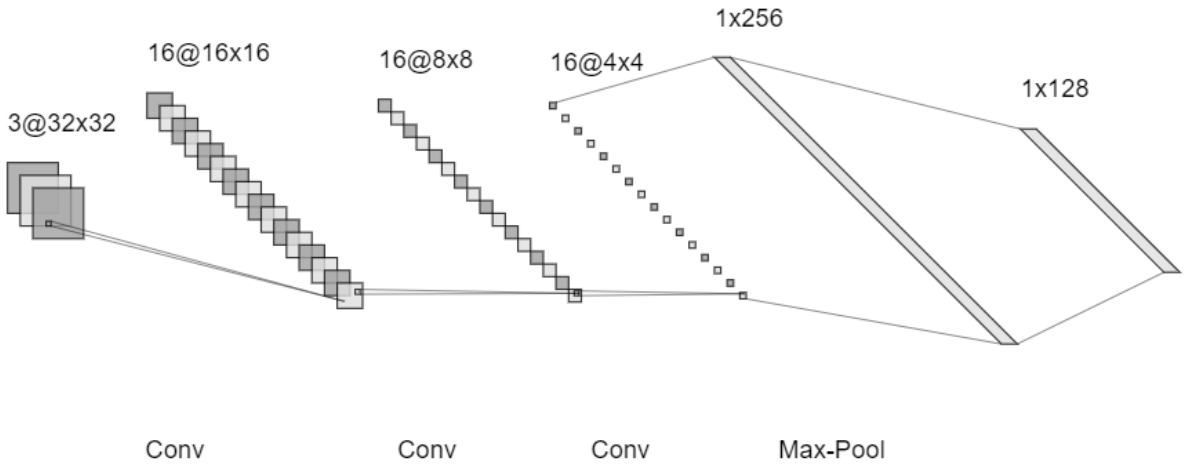

Figure 4.3. NuNet Model Architecture with Three Layers

4.2 NuNet Feature Extraction

Features extracted from the first, second and third layers are presented in figures 4.4, 4.5 and 4.6 respectively for the "doNotEnter" traffic sign. Figure 4.4 shows 16 features extracted from the input image by the first convolutional layer of the model. For very distinct images, this first layer can accurately classify images at high accuracy, however, for distorted images or similar images, it becomes challenging for the model to only use one layer to get the best results. In such an instance, additional convolutional layers are added to improve the accuracy and reduce the error of classification. This addition is mostly done at later epochs during training.

Figures 4.5 represents features extracted from the second layer of the adaptive model that gets added in the event that another layer needs to be employed to enhance model performance. If the model's performance or error reduction stalls, another layer is added which features are presented in figure 4.6. The more layers are added, the slower the model performs; however, the training would have been at a later epoch before extra layers may be required. The maximum number of layers the model can add may be capped at the discretion of the developer; this research

used 3 layers. The extra features the model extracts help the model learn important details that it might have missed in previous epochs. This incremental layering is the main idea behind NuNet's architecture.

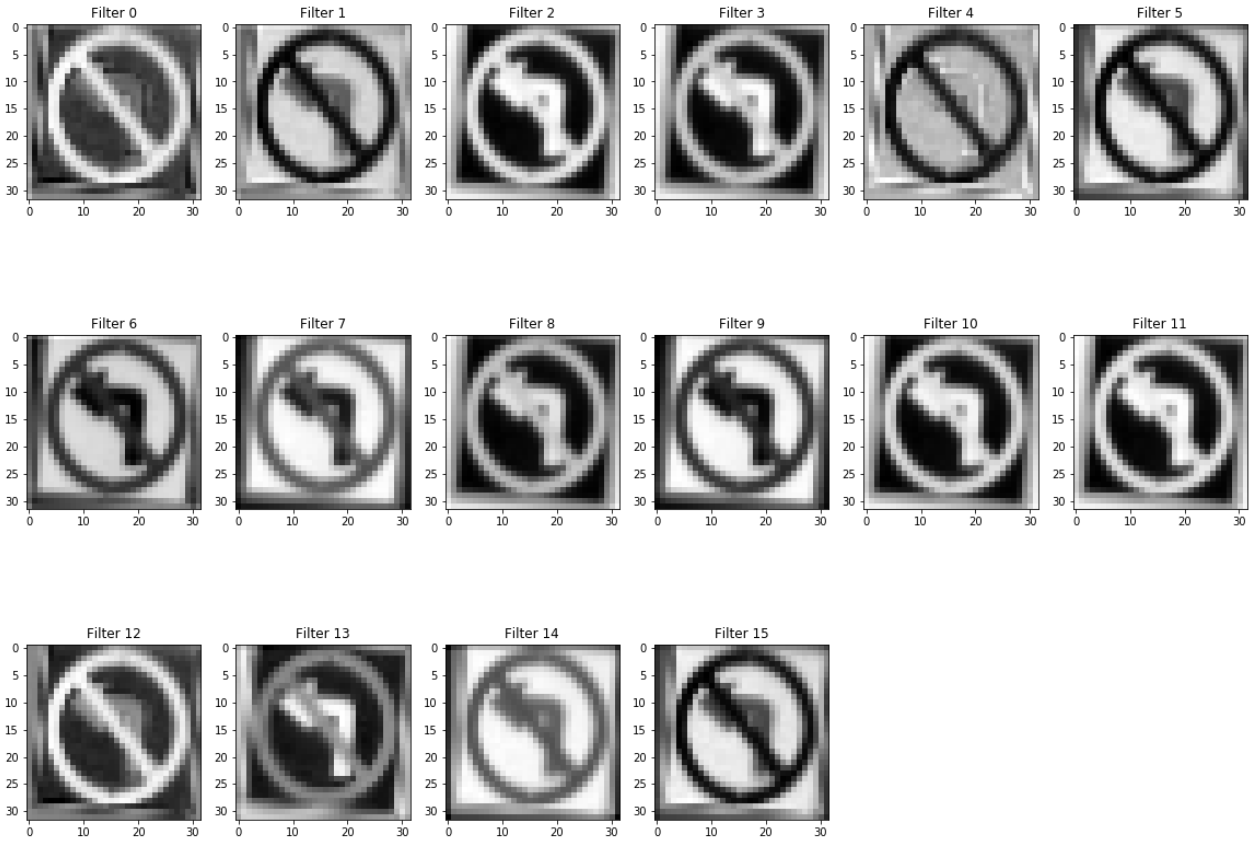

Figure 4.4. Features Extracted from the First Layer of the NuNet Model

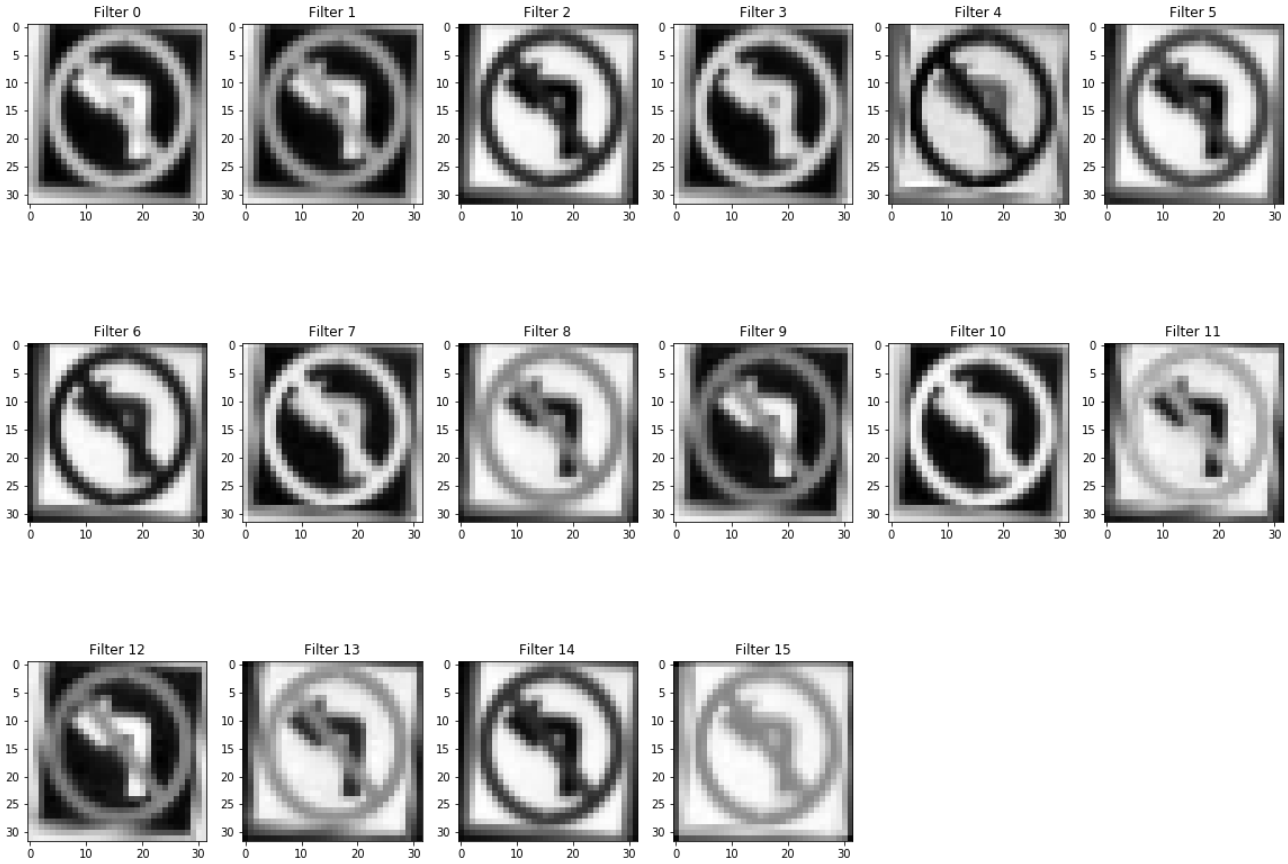

Figure 4.5. Features Extracted from the Second Layer of the NuNet Model

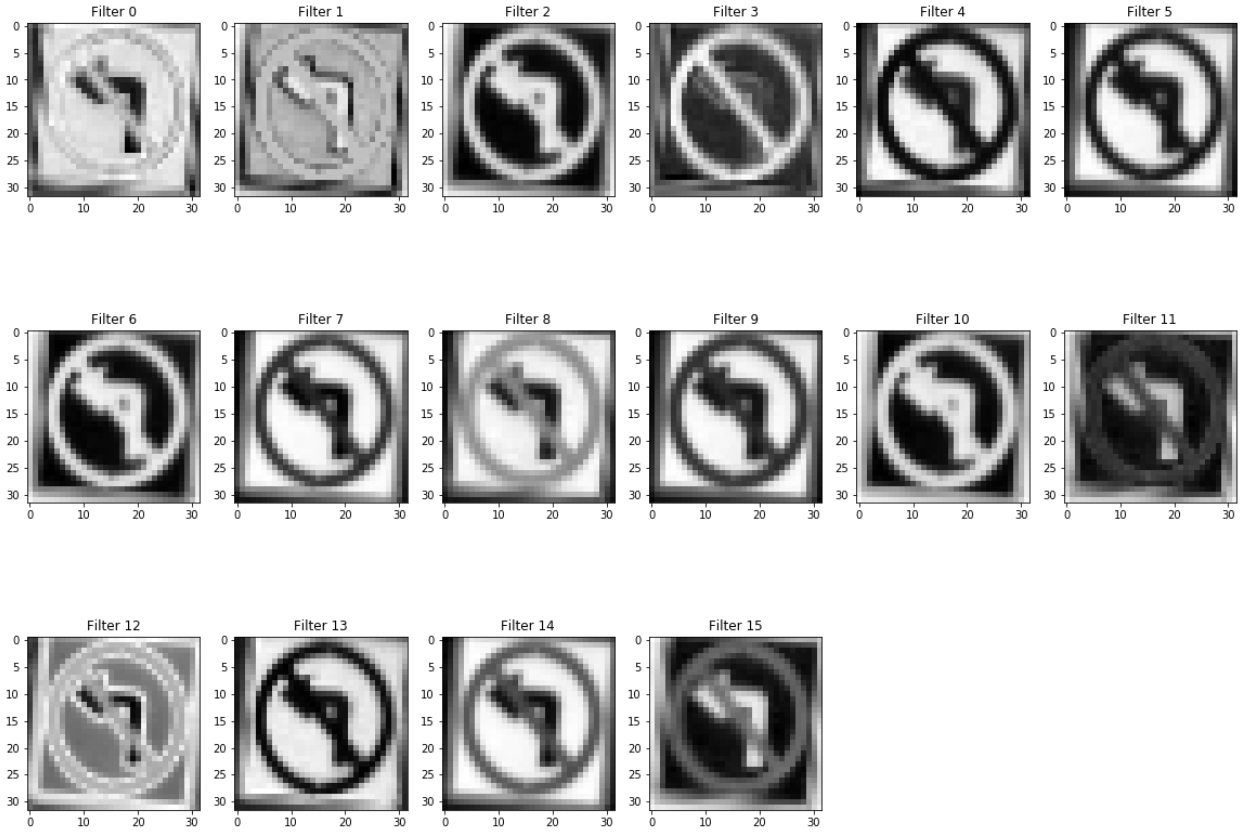

Figure 4.6. Features Extracted from the Third Layer of the NuNet Model

4.3 NuNet Results on LISA Dataset

The results from training the NuNet model on the LISA dataset are presented in this section. The training was carried out for two splits, the first is 80/20 for training and validation sets respectively. The second split is an inversion of the first; using the 20% split for training and the 80% split for validation. The results from the first (80/20) split are presented in section 4.3.1 and the results from the second split are presented in section 4.3.2.

4.3.1 Result Training NuNet on LISA Dataset

With an 80/20 split, the NuNet model took approximately 10 hours to train on the LISA dataset (introduced in section 3.4) for 600 epochs. After the 600th epoch, the training accuracy recorded for NuNet was 99.73%. The validation accuracy of NuNet was 99.83% is the LISA

dataset. The training was smooth and did not show any irregularities. Figure 4.7 shows the accuracies plot recorded for the LISA training and validation sets. Figure 4.8 shows the losses plot for the LISA training and validation sets. The loss recorded for NuNet trained on LISA was 7.9% for the training set and 9.6% for the validation set for this split ratio.

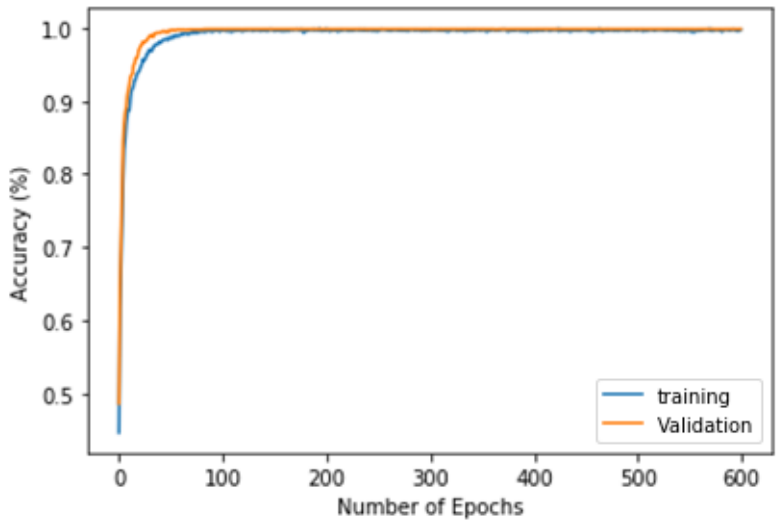

Figure 4.7. A plot of Training and Validation Accuracies for NuNet on LISA

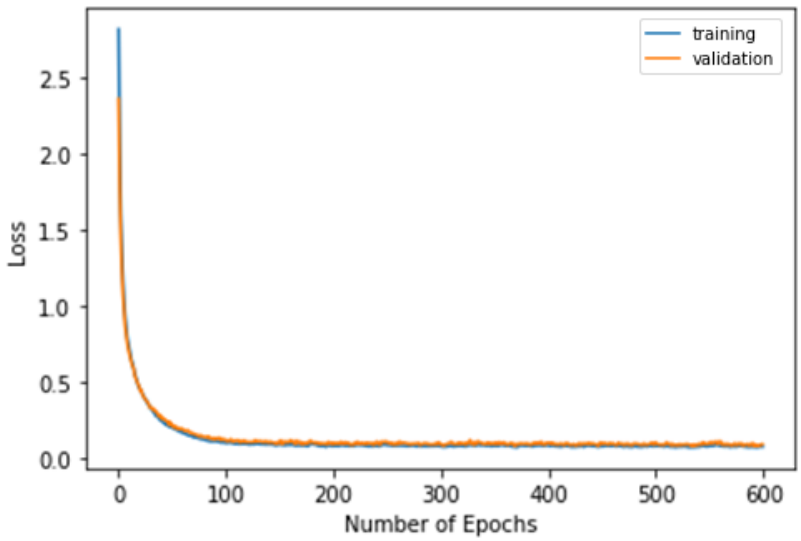

Figure 4.8. A Plot of Training and Validation sets Losses for NuNet on LISA

Figure 4.9 presents the confusion matrix for the validation set of the LISA dataset. Intermediate confusion matrices were plotted during training and presented in appendix C. Only 3 samples of the "speedLimit35" class in the validation set were misclassified as "pedestrianCrossing".

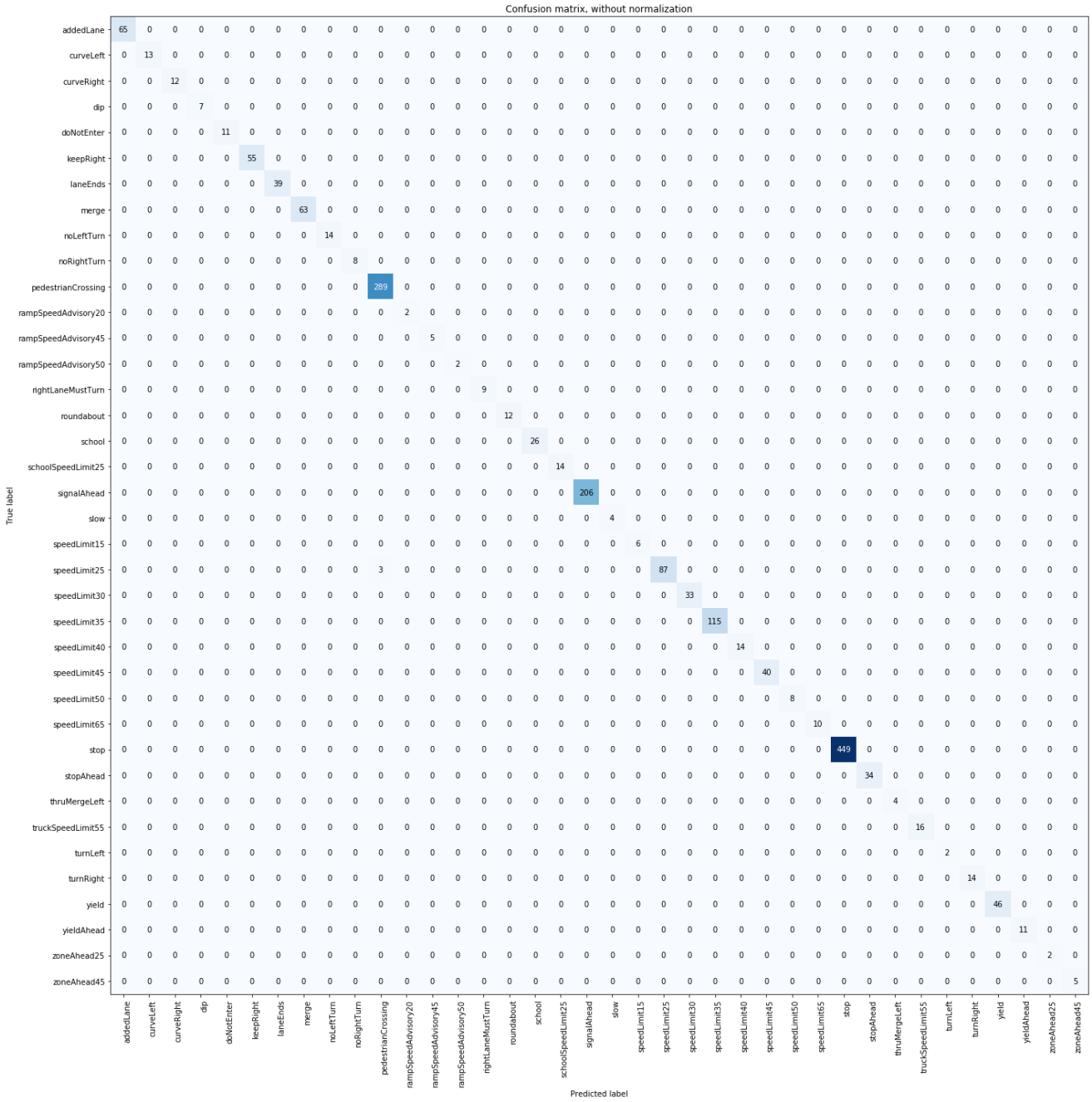

Figure 4.9. Confusion Matrix of NuNet Model on LISA Dataset

4.4 NuNet Results on CIB TS V1 Dataset

The results from training the NuNet model on the CIB dataset are presented in this section. Here, a second split, an inversion of the original 80/20 split is used for training. The results of the original split are presented in section 4.4.1 and the results of the 20/80 split are also presented in section 4.4.2.

4.4.1 Result Training NuNet on the CIB Dataset

The model performs at an accuracy of 100% on the training set and records a validation accuracy of 100% on the CIB dataset. An error of 1.8% was recorded for the training samples and a validation error of 1.9% was recorded. The model generalizes well and does not underfit or overfit the dataset. Training took approximately 3 hours; a little less than a third the time it took to train VGGNet on the same dataset. The accuracy plots for training and validation sets are shown in figure 4.10. Training plateaued within the first 30 epochs. Figure 4.11 presents loss plots for the training and validation sets.

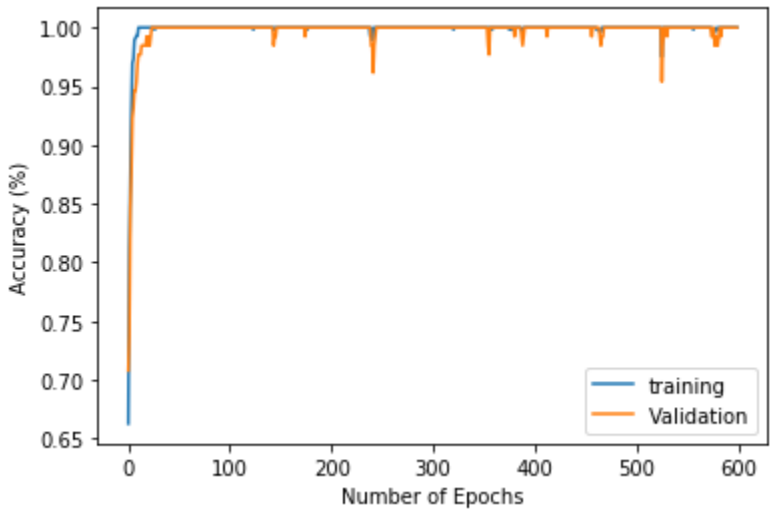

Figure 4.10. A Plot of Training and Validation Accuracies for the CIB Dataset

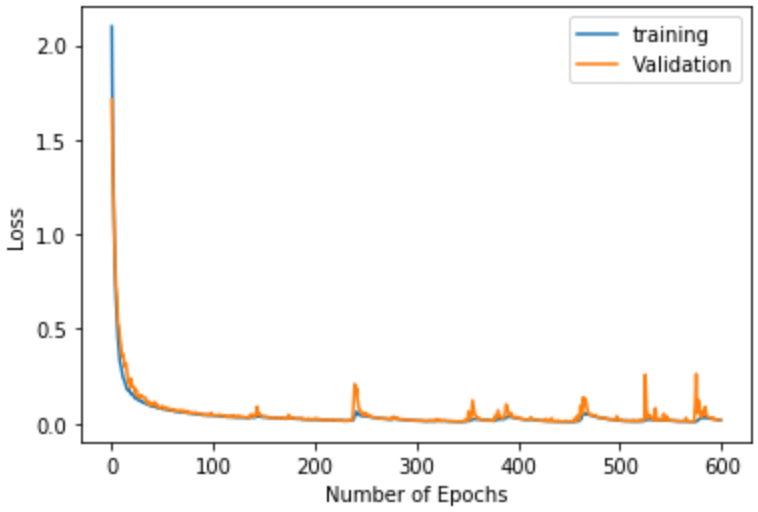

Figure 4.11. A Plot of Training and Validation Losses for the CIB Dataset

The model was trained over 600 epochs and provides more reliable training and validation than was encountered with VGGNet in section 3.6. The NuNet model performs well on smaller datasets and trains faster. Figure 4.12 shows the confusion matrix for the validation set. Intermediate confusion matrices were plotted and presented in appendix D.

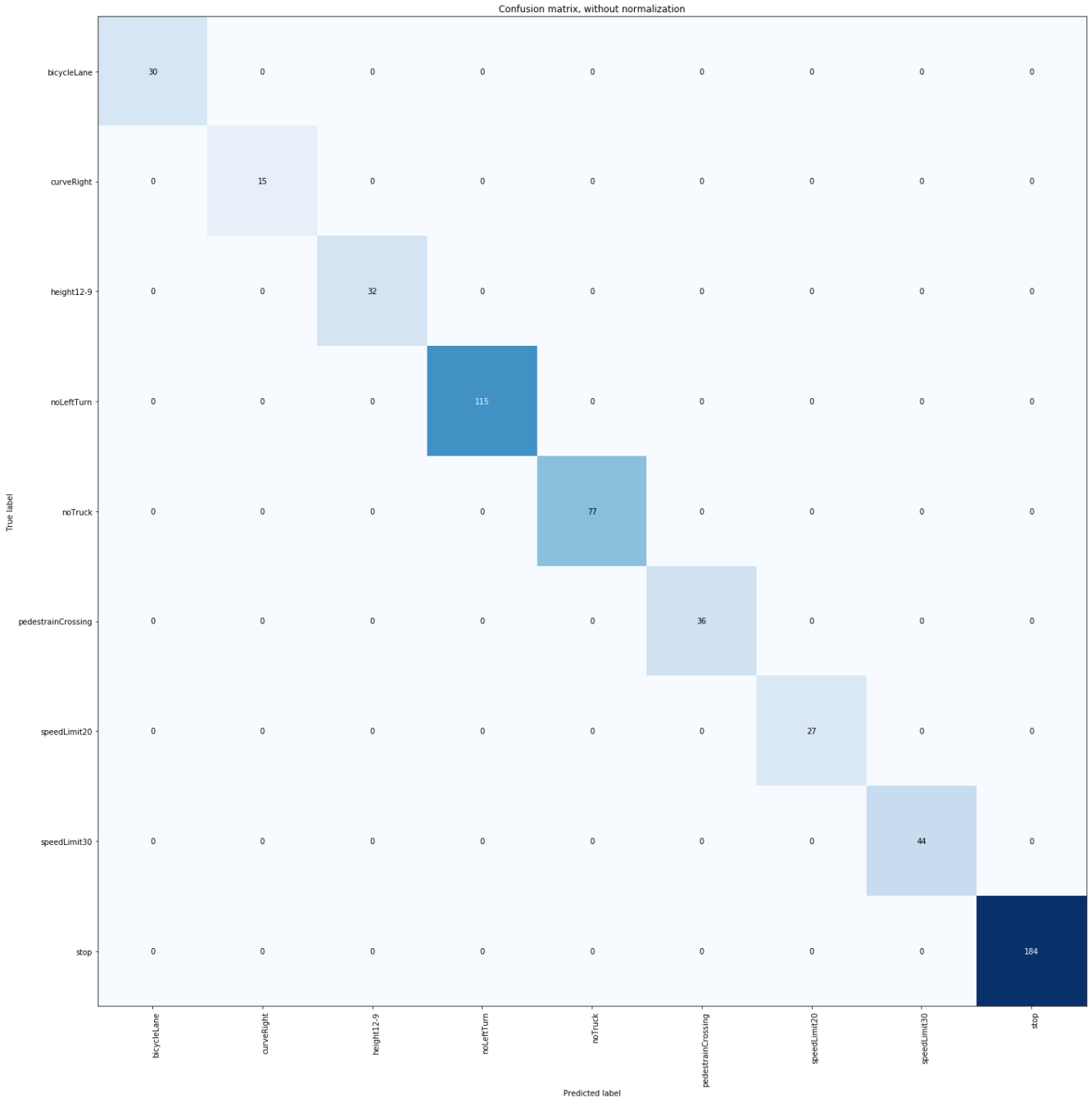

Figure 4.12. Confusion Matrix of Model after Training on CIB Dataset

4.4.2 Result Training NuNet on the 20/80 CIB Dataset Split

NuNet performed at accuracies of 100% and recorded an error of 3.1% for both the training and validation sets. The error in this split was higher than the previous split, which shows that

training with a smaller dataset can affect the error recorded; however, training is smoother than on the 80/20 split. Accuracy and loss plots for the results are presented in figures 4.13 and 4.14 for this split.

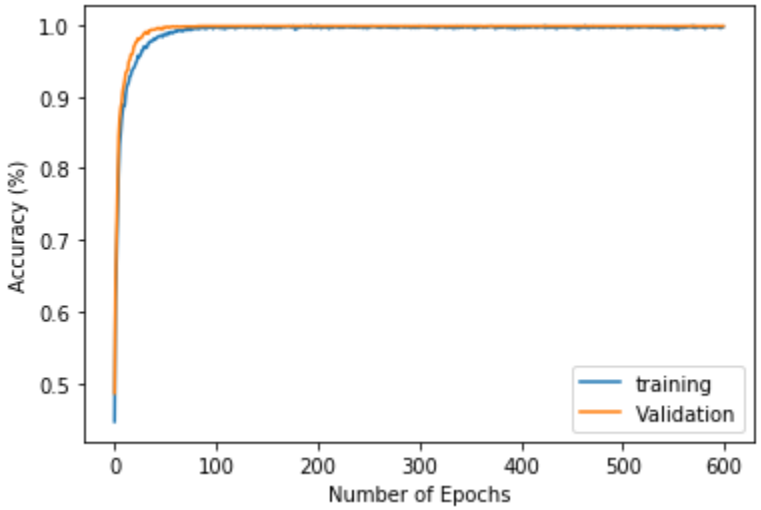

Figure 4.13. A Plot of Training and Validation Accuracies for CIB Dataset split at 20/80

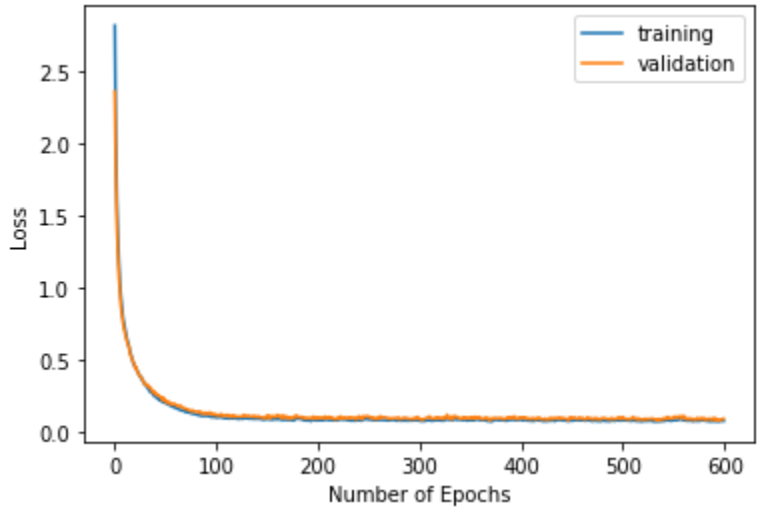

Figure 4.14. A Plot of Training and Validation Losses for CIB Dataset split at 20/80

4.5 Discussion

Table 4.1 compares the results of NuNet and VGGNet based on the number of layers, training time, and model performance for both LISA TS and CIB TS-V1 datasets. The results are also compared against that of an SVM classifier. SVM overfits the model with 99.68% training accuracy and 83.02% validation accuracy on LISA. SVM also overfits on CIB with 100% training accuracy and 82.31% validation accuracy. NuNet recorded an accuracy of 100% for both training and validation sets on CIB and recorded 99.73% and 99.83% respectively on LISA. VGGNet recorded the best training accuracy for LISA at 99.93% and ties NuNet for the validation accuracy. VGGNet overfits on CIB with training and validation accuracy of 100% and 96.92% respectively. NuNet recorded the lowest error on CIB with 1.8% and 1.9% on the training and validation sets respectively. It recorded an error of 7.9% on the training set and 9.6% on the validation set of LISA. Whereas VGGNet recorded an error of 6.6% on the training set and 7.1 % on the validation set of LISA. VGGNet recorded an error of 1.8% and 5.6% for training and validation sets of CIB.

Table 4.1. Comparison between SVM, NuNet, and VGGNet trained on LISA and CIB Datasets

	SVM		**NuNet**		**VGGNet**	
	LISA	**CIB**	**LISA**	**CIB**	**LISA**	**CIB**
No. of Layers			Variable		Fixed	
~ Training Time	12 min	1 min	10 hours	**3 hours**	26 hours	9 hours
Train Accuracy	99.68%	100%	99.73%	**100%**	**99.93%**	100%
Valid. Accuracy	83.02%	82.31%	**99.83%**	**100%**	**99.83%**	96.92%
Training Loss			7.9%	**1.8%**	**6.6%**	**1.8%**
Validation Loss			9.6%	**1.9%**	**7.1%**	**5.6%**

CHAPTER 5

Conclusion and Future Directions

ADAS help drivers with little things that can escape their attention. The issue of traffic sign recognition derives its roots from the fact that certain signs, while relevant to be noticed by humans, may go unnoticed because of the noise that might make it indistinctive, or just that humans selectively filter things they see based on preconceived mindset. A computer is rather trained to pick up things that the human eye may overlook. Traffic sign recognition on U.S. datasets has not been thoroughly researched. This research introduces a deep learning technique for recognizing U.S. traffic signs.

VGGNet obtained an accuracy of 99.93% and 99.83%, respectively on training and validation samples on the LISA dataset. The NuNet model performs at a similar accuracy of 99.73% and 99.83% for training and validation sets, respectively. The VGGNet, however, performs poorly on the CIB dataset at an accuracy of 100% on the training set and 96.92% on the validation set. On the CIB dataset, NuNet performs at an accuracy of 100% on the training set and 100% on the validation set. The error recorded for VGGNet on LISA was 6.6% and 7.1% respectively for training and validation sets. The error recorded for VGGNet on CIB was 1.8% and 5.6% respectively for training and validation sets.

The NuNet also trains faster than VGGNet as it only extracts features, necessary for recognizing the traffic sign images. Hence, if the model is able to use fewer features to recognize a traffic sign, the model does not extract more features to recognize that specific traffic signs as this results in the model being slower. VGGNet also used approximately 9 hours for training on the CIB dataset while NuNet used approximately 3 hours. On the LISA dataset, VGGNet trained for approximately 26 hours whiles NuNet trained for approximately 10 hours.

A model that is able to adjust its parameters has therefore shown to be a powerful tool for training a deep neural network than one which does not. NuNet is able to train on fewer data than most deep models without overfitting because it does not extract unnecessary information. However, it is able to increase the number of layers to classify large and complex datasets.

Future work will be concerned with conducting experiments to investigate automating the number of epochs the model has to be trained for. This will consist of defining a decay function that will allow the model to continue training as long as improvements can be made to the model's performance on a specific dataset. In this case, one of two things would be proposed: 1.) in a bottom-up manner, train the model by setting a conditional statement controlled by a progression function; 2.) in a top-down approach, set a maximum number of epochs for training and decrease this number by a decay function. This proposed approach to stop training rather than setting a constant number of epochs for training will allow the model to self-pace training.

An investigation would also be conducted into how the model can automatically select the number of layers for training. Currently, the model can only add layers if a condition is satisfied. Future work would consider the model adding or removing layers during training to further enhance performance and probably speed. The model would be tested on multiple datasets to test its suitability towards any object recognition task.

A proposed strategy for deciding an epoch to stop training would be looked into. The term training decay factor, which represents a constant would be researched for automatically choosing a stopping epoch during training. The proposal states that once the model reaches a global maximum, let's say of 99%, the number of epochs remaining is decayed by subtracting the decay factor from it. This factor will be a great component if it can be modeled into a function. In that case, the model can train itself without being given a set number of epochs to train over, but rather,

it will stop when the function's condition is satisfied. This will imply that the epoch selection will follow a bottom-up approach, where training starts from epoch one, to epoch n, with n being decided on by the training decay function. Another way, similar to the regular way of training is to set a number of epochs for training and let the function decay the remaining epochs when the condition set by the function is met in a top-down approach.

References

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G.S., Davis, A., Dean, J., Devin, M. and Ghemawat, S., (2016). Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*.

Arcos-Garcia, A., Alvarez-Garcia, J. A., & Soria-Morillo, L. M. (2018). Evaluation of deep neural networks for traffic sign detection systems. *Neurocomputing*, *316*, 332-344.

Berkaya, S. K., Gunduz, H., Ozsen, O., Akinlar, C., & Gunal, S. (2016). "On circular traffic sign detection and recognition". *Expert Systems with Applications, 48*, pp. 67-75.

Cireşan, D., Meier, U., Masci, J., and Schmidhuber, J. (2011). "A Committee of Neural Networks for Traffic Sign Classification". *The 2011 International Joint Conference on Neural Networks*, San Jose, CA, 2011, pp. 1918-1921.

Cireşan, D., Meier, U., & Schmidhuber, J. (2012). "Multi-Column Deep Neural Networks for Image Classification". *arXiv preprint arXiv:1202.2745*.

Escalera, A. D. L., Moreno, L., Salichs, M. A., & Armingol, J. M. (1997). Road traffic sign detection and classification.

Fang, C. Y., Fuh, C. S., Yen, P. S., Cherng, S., & Chen, S. W. (2004). "An Automatic Road Sign Recognition System Based on a Computational Model of Human Recognition Processing". *Computer vision and Image understanding*, 96(*2*), 237-268.

Girshick, R. (2015). Fast R-CNN. In *Proceedings of the IEEE international conference on computer vision* (pp. 1440-1448).

Huang, G. B., Zhu, Q. Y., & Siew, C. K. (2006) "Extreme Learning Machine: Theory and Applications." *Neurocomputing*, 70(*1*), 489-501.

Huang, Z., Yu, Y., Gu, J., & Liu, H. (2017). “An Efficient Method for Traffic Sign Recognition Based on Extreme Learning Machine”. IEEE Transactions on Cybernetics, 47(*4*), 920-933.

Huval, B., Wang, T., Tandon, S., Kiske, J., Song, W., Pazhayampallil, J., Andriluka, M., Rajpurkar, P., Migimatsu, T., Cheng-Yue, R., Mujica, F., Coates, A., & Ng, Y., A., (2015). “An Empirical Evaluation of Deep Learning on Highway Driving”. *arXiv preprint arXiv:1504.01716*.

Jin, J., Fu, K., and Zhang, C. (2014). "Traffic Sign Recognition with Hinge Loss Trained Convolutional Neural Networks". In *IEEE Transactions on Intelligent Transportation Systems*, *vol. 15*, no. 5, pp. 1991-2000, Oct. 2014.

Jurišić, F., Filković, I., & Kalafatić, Z. (2015). “Multiple-Dataset Traffic Sign Classification with OneCNN”. In *Pattern Recognition (ACPR), 2015 3rd IAPR Asian Conference on* (pp. 614-618). IEEE.

Kluyver, T., Ragan-Kelley, B., Pérez, F., Granger, B.E., Bussonnier, M., Frederic, J., Kelley, K., Hamrick, J.B., Grout, J., Corlay, S. and Ivanov, P. (2016). Jupyter Notebooks-a publishing format for reproducible computational workflows. In *ELPUB* (pp. 87-90).

Laguna, R., Barrientos, R., Blázquez, L. F., & Miguel, L. J. (2014). “Traffic Sign Recognition Application Based on Image Processing Techniques”. *IFAC Proceedings Volumes*, 47(*3*), 104-109.

Larsson, F., Felsberg, M., & Forssen, P. E. (2011). “Correlating Fourier Descriptors of Local Patches for Road Sign Recognition”. *IET Computer Vision*, 5(*4*), 244-254.

Larsson, F., & Felsberg, M. (2011). Using Fourier descriptors and Spatial Models for Traffic Sign Recognition. In the *Scandinavian Conference on Image Analysis* (pp. 238-249). Springer, Berlin, Heidelberg.

LeCun, Y., Bengio, Y., & Hinton, G. (2015). “Deep Learning”. *Nature*, 521(*7553*), 436.

Li, J., Mei, X., Prokhorov, D., & Tao, D. (2017). “Deep Neural Network for Structural Prediction and Lane Detection in Traffic Scene”. *IEEE Transactions on Neural Networks and Learning Systems*, 28(*3*), 690-703.

Li, Y., Møgelmose, A., & Trivedi, M. M. (2016). “Pushing the “Speed Limit”: High-Accuracy US Traffic Sign Recognition with Convolutional Neural Networks”. *IEEE Transactions on Intelligent Vehicles*, 1(*2*), 167-176.

Luo, P., Tian, Y., Wang, X., & Tang, X. (2014). “Switchable Deep Network for Pedestrian Detection”. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 899-906).

Maldonado-Bascón, S., Lafuente-Arroyo, S., Gil-Jimenez, P., Gómez-Moreno, H., & López-Ferreras, F. (2007). “Road-sign Detection and Recognition Based on Support Vector Machines”. *IEEE transactions on intelligent transportation systems*, 8(*2*), 264-278.

Matas, J., Chum, O., Urban, M., & Pajdla, T. (2004). “Robust Wide-Baseline Stereo from Maximally Stable Extremal Regions”. *Image and Vision Computing*, 22(*10*), 761-767.

Mathias, M., Timofte, R., Benenson, R., & Van Gool, L. (2013). “Traffic Sign Recognition—How Far Are We from The Solution?”. In *Neural Networks (IJCNN), The 2013 International Joint Conference* on (pp. 1-8). IEEE.

Moutarde, F., Bargeton, A., Herbin, A., & Chanussot, L. (2007). "Robust on-vehicle real-time Visual Detection of American and European Speed Limit Signs, with a Modular Traffic Signs Recognition System". In *IEEE Intelligent Vehicles Symposium*, Istanbul, 2007, pp. 1122-1126.

Nuakoh, E. B., Roy, K., Yuan, X., & Esterline, A. (2019, July). Deep Learning Approach for US Traffic Sign Recognition. In *Proceedings of the 2019 3rd International Conference on Deep Learning Technologies* (pp. 47-50). ACM.

Ouyang, W., & Wang, X. (2013). "Joint Deep Learning for Pedestrian Detection". In *Proceedings of the IEEE International Conference on Computer Vision* (pp. 2056-2063).

Ouyang, W., & Wang, X. (2012). "A discriminative Deep Model for Pedestrian Detection with Occlusion Handling". In *2012 IEEE Conference on Computer Vision and Pattern Recognition* (pp. 3258-3265). IEEE.

Sermanet, P., & LeCun, Y. (2011). "Traffic Sign Recognition with Multi-Scale Convolutional Networks". In *Neural Networks (IJCNN), The 2011 International Joint Conference* on (pp. 2809-2813). IEEE.

Sermanet, P., Chintala, S., & LeCun, Y. (2012). "Convolutional Neural Networks Applied to House Numbers Digit Classification". In *Pattern Recognition (ICPR), 2012 21st International Conference* on (pp. 3288-3291). IEEE.

Sermanet, P., Eigen, D., Zhang, X., Mathieu, M., Fergus, R., & LeCun, Y. (2013a). "Overfeat: Integrated Recognition, Localization and Detection Using Convolutional Networks". *arXiv preprint arXiv:1312.6229*.

Sermanet, P., Kavukcuoglu, K., Chintala, S., & LeCun, Y. (2013b). "Pedestrian Detection with Unsupervised Multi-Stage Feature Learning". In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 3626-3633).

Schmidhuber, J. (2015). "Deep Learning in Neural Networks: An Overview". *Neural Networks, 61,* 85-117.

Shustanov, A., & Yakimov, P. (2017). “CNN Design for Real-Time Traffic Sign Recognition”. *Procedia Engineering*, *201*, 718-725.

Stallkamp, J., Schlipsing, M., Salmen, J., & Igel, C. (2011). “The German traffic sign recognition benchmark: a multi-class classification competition”. In *Neural Networks (IJCNN), The 2011 International Joint Conference* on (pp. 1453-1460). IEEE.

Stallkamp, J., Schlipsing, M., Salmen, J., & Igel, C. (2012). “Man vs. computer: Benchmarking Machine Learning Algorithms for Traffic Sign Recognition”. *Neural Networks, Volume 32*, 2012, Pages 323-332.

Tian, Y., Luo, P., Wang, X., & Tang, X. (2015). “Pedestrian Detection Aided by Deep Learning Semantic Tasks”. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 5079-5087).

Timofte, R., Zimmermann, K., & Van Gool, L. (2014). “Multi-View Traffic Sign Detection, Recognition, and 3D Localization”. *Machine vision and applications*, 25(*3*), 633-647.

Torresen, J., Bakke, J. W., & Sekanina, L. (2004). "Efficient Recognition of Speed Limit Signs". *Proceedings. The 7th International IEEE Conference on Intelligent Transportation Systems (IEEE Cat. No.04TH8749)*, Washington, WA, USA, 2004, pp. 652-656.

Tzutalin. LabelImg. Git code (2015). https://github.com/tzutalin/labelImg.

Youssef, A., Albani, D., Nardi, D., & Bloisi, D. D. (2016). “Fast Traffic Sign Recognition Using Color Segmentation and Deep Convolutional Networks”. In *International Conference on Advanced Concepts for Intelligent Vision Systems* (pp. 205-216). Springer, Cham.

Yu, Y., Li, J., Wen, C., Guan, H., Luo, H., & Wang, C. (2016). Bag-of-Visual-Phrases and Hierarchical Deep Models for Traffic Sign Detection and Recognition in Mobile Laser Scanning Data. *ISPRS Journal of Photogrammetry and Remote Sensing*, *113*, 106-123.

Zhu, Z., Liang, D., Zhang, S., Huang, X., Li, B., & Hu, S. (2016). “Traffic-Sign Detection and Classification in The Wild”. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 2110-2118).

Appendix A

Appendix A presents the recorded values during training for losses and accuracies of training and validation sets for experiments using VGGNet on the LISA dataset in section 3.5.1.

Table A.2. Loss and Accuracy Values for Training and Validation

Epoch	Training Loss	Validation Loss	Training Accuracy	Validation Accuracy
1	3.097761708	2.812614	0.257746	0.25584
2	2.74759705	2.611326	0.37	0.380057
3	2.612055893	2.425479	0.384789	0.380627
4	2.402351074	2.249338	0.419859	0.435328
5	2.152300956	1.993753	0.478592	0.491168
6	1.941277798	1.860928	0.534225	0.538462
7	1.76022558	1.637012	0.616761	0.617094
8	1.632121012	1.452343	0.663944	0.678063
9	1.432769911	1.305922	0.719859	0.717379
10	1.263072603	1.16588	0.759577	0.757265
11	1.128576598	1.064556	0.778732	0.791453
12	1.004881547	0.958055	0.822676	0.825641
13	0.930744597	0.888187	0.835634	0.847863
14	0.834876621	0.777148	0.861972	0.866667
15	0.759428133	0.736702	0.87662	0.887749
16	0.709209877	0.669678	0.899859	0.896296

17	0.633351619	0.610647	0.912817	0.9151
18	0.563159542	0.54721	0.928873	0.938462
19	0.513774723	0.507296	0.940423	0.950997
20	0.473833224	0.482398	0.945352	0.958974
21	0.445526152	0.412484	0.959155	0.969801
22	0.414989634	0.411033	0.964507	0.97094
23	0.379712744	0.376298	0.968873	0.976638
24	0.364880943	0.362073	0.968732	0.981766
25	0.338104381	0.340451	0.97507	0.982906
26	0.324289165	0.315778	0.97831	0.986895
27	0.309020131	0.300437	0.982958	0.988034

28	0.295691185	0.298011	0.984366	0.988034
29	0.279663329	0.286916	0.982535	0.989174
30	0.267640402	0.279588	0.986338	0.989744
31	0.270089912	0.266089	0.98338	0.989744
32	0.25960971	0.261388	0.988732	0.992023
33	0.241157197	0.257769	0.987042	0.993732
34	0.240324123	0.247032	0.991972	0.990883
35	0.223828847	0.248493	0.990704	0.993162
36	0.213103104	0.222894	0.994085	0.994302

37	0.212142226	0.226345	0.991408	0.995442
38	0.202207726	0.223341	0.994648	0.993732
39	0.212327417	0.223873	0.994648	0.995442
40	0.19783907	0.21376	0.99507	0.994302
41	0.19226613	0.201062	0.996761	0.992023
42	0.196465741	0.217418	0.994085	0.994302
43	0.189183241	0.194391	0.994507	0.995442
44	0.179704767	0.204941	0.994648	0.994302
45	0.170688549	0.185784	0.996761	0.994872
46	0.164913765	0.179836	0.997465	0.994302
47	0.16107415	0.177198	0.997042	0.995442
48	0.157601433	0.167722	0.997887	0.995442
49	0.153525364	0.168045	0.997465	0.996581
50	0.149118491	0.167952	0.997887	0.997721
51	0.144126812	0.166545	0.998592	0.997721
52	0.141357999	0.158834	0.998451	0.997151
53	0.1387262	0.161264	0.998169	0.997151
54	0.137388932	0.155557	0.996901	0.996581
55	0.137729378	0.14896	0.997746	0.998291
56	0.141356515	0.152151	0.995915	0.996011
57	0.134909629	0.151925	0.998592	0.997721
58	0.13071579	0.141499	0.998592	0.997151

59	0.131283669	0.140071	0.998028	0.997721
60	0.130470107	0.14104	0.998451	0.996581
61	0.13486209	0.151188	0.995915	0.995442
62	0.134527999	0.156826	0.997042	0.996581
63	0.126976339	0.138246	0.997746	0.997721
64	0.121351666	0.133655	0.998873	0.998291
65	0.11567679	0.132374	0.999577	0.998291
66	0.117056048	0.125481	0.997746	0.996581
67	0.114088887	0.120939	0.998451	0.998291
68	0.11234433	0.118715	0.999718	0.998291
69	0.111003019	0.130121	0.998169	0.998291
70	0.107394829	0.11871	0.998592	0.998291
71	0.108562823	0.117451	0.999577	0.998291
72	0.104883381	0.116203	0.999014	0.997721
73	0.102643461	0.109029	0.998873	0.998291
74	0.106667563	0.110575	0.997746	0.997151
75	0.105868465	0.120043	0.998028	0.998291
76	0.113847493	0.124829	0.997042	0.998291
77	0.108408039	0.128345	0.998028	0.997721
78	0.110094978	0.126199	0.997746	0.997151
79	0.102781496	0.115095	0.998169	0.997721
80	0.104413322	0.119681	0.997183	0.997721

81	0.110846137	0.126486	0.997183	0.998291
82	0.106055579	0.115613	0.998028	0.997721
83	0.103272441	0.111051	0.998028	0.997721
84	0.102868141	0.114027	0.999155	0.998291
85	0.102510117	0.113878	0.999155	0.998291
86	0.102762544	0.12357	0.997465	0.997721
87	0.100402635	0.105502	0.997887	0.998291
88	0.100132396	0.113936	0.997887	0.997721
89	0.103828356	0.116034	0.997324	0.997151
90	0.107987113	0.12558	0.996338	0.998291
91	0.103258746	0.113905	0.998592	0.998291
92	0.101762934	0.111821	0.998451	0.998291
93	0.097466391	0.109025	0.997606	0.998291
94	0.100628683	0.109401	0.997042	0.998291
95	0.094872501	0.107419	0.998873	0.982336
96	0.090064796	0.104649	0.999014	0.998291
97	0.092836177	0.110506	0.997746	0.998291
98	0.091442583	0.097661	0.999014	0.998291
99	0.092570731	0.112891	0.99831	0.997721
100	0.08948961	0.109827	0.997746	0.998291
101	0.089143257	0.103652	0.998451	0.998291
102	0.096879486	0.108767	0.996901	0.997721

103	0.095775502	0.102507	0.998592	0.998291
104	0.089580127	0.102325	0.998873	0.998291
105	0.085332424	0.09662	0.998169	0.997151
106	0.088012294	0.091967	0.998592	0.998291
107	0.089153317	0.109024	0.998873	0.983476
108	0.088218677	0.111261	0.998169	0.995442
109	0.083477548	0.091369	0.999014	0.998291
110	0.085715038	0.10246	0.998028	0.998291
111	0.084544391	0.105286	0.997746	0.997721
112	0.083239266	0.098089	0.997465	0.997721
113	0.092902807	0.104665	0.996479	0.998291
114	0.085067838	0.100434	0.997887	0.997721
115	0.083951141	0.10477	0.996338	0.997721
116	0.083170018	0.097522	0.997887	0.997721
117	0.080025426	0.090901	0.998873	0.997721
118	0.084274337	0.08735	0.998732	0.997721
119	0.081533764	0.093792	0.999859	0.998291
120	0.083590259	0.115519	0.994366	0.997721
121	0.088640079	0.095787	0.997887	0.998291
122	0.085462135	0.091281	0.998592	0.998291
123	0.086426533	0.09912	0.998873	0.997721
124	0.084938435	0.099534	0.998169	0.998291

125	0.090667325	0.100691	0.998592	0.998291
126	0.081207441	0.095752	0.998592	0.997721
127	0.083045874	0.100969	0.998169	0.997721
128	0.081756009	0.119785	0.996056	0.983476
129	0.079671224	0.092432	0.997746	0.998291
130	0.084739199	0.088527	0.999296	0.998291
131	0.079758419	0.090257	0.999577	0.998291
132	0.074886658	0.088462	0.999437	0.998291
133	0.076498982	0.092746	0.998732	0.998291
134	0.075882708	0.087965	0.999296	0.998291
135	0.071367288	0.089318	0.999437	0.998291
136	0.070490729	0.083398	0.999014	0.998291
137	0.071217467	0.085095	0.998169	0.997721
138	0.071989853	0.091423	0.998028	0.997721
139	0.073233423	0.09088	0.998169	0.997721
140	0.07414435	0.09128	0.999014	0.998291
141	0.073933004	0.080099	0.999437	0.998291
142	0.072827883	0.096145	0.997887	0.997151
143	0.073828028	0.091136	0.998169	0.998291
144	0.096089865	0.105974	0.995634	0.997151
145	0.087503672	0.10099	0.996056	0.998291
146	0.086745189	0.098305	0.997324	0.997151

147	0.090264101	0.098288	0.997746	0.998291
148	0.095267341	0.105012	0.997465	0.998291
149	0.084808027	0.101358	0.999577	0.998291
150	0.082648386	0.088497	0.999437	0.998291
151	0.082515598	0.104704	0.998028	0.983476
152	0.093445161	0.115006	0.997465	0.998291
153	0.095819709	0.134853	0.993803	0.997721
154	0.093356526	0.104543	0.998169	0.998291
155	0.092690022	0.112187	0.997324	0.997151
156	0.105265884	0.119023	0.997746	0.997151
157	0.093150276	0.105438	0.99831	0.998291
158	0.085865485	0.103989	0.999155	0.997721
159	0.084260299	0.09156	0.999437	0.998291
160	0.080926149	0.095875	0.99831	0.998291
161	0.081580831	0.087464	0.999155	0.998291
162	0.079053357	0.086656	0.999437	0.998291
163	0.078466784	0.098668	0.998028	0.997151
164	0.077352652	0.092606	0.999155	0.998291
165	0.077405458	0.084284	0.998732	0.998291
166	0.076364515	0.081207	0.999155	0.998291
167	0.077878133	0.092396	0.999014	0.998291
168	0.072653512	0.078241	0.999718	0.998291

169	0.071105569	0.078374	0.999577	0.998291
170	0.07292468	0.08376	0.998451	0.998291
171	0.077272317	0.088885	0.99662	0.997151
172	0.08124128	0.086807	0.999437	0.998291
173	0.076449325	0.082665	0.999014	0.998291
174	0.077671162	0.084931	0.998873	0.998291
175	0.082476996	0.093815	0.996338	0.995442
176	0.087355349	0.098646	0.999014	0.998291
177	0.084597205	0.093179	0.999296	0.998291
178	0.086357427	0.104309	0.99831	0.997721
179	0.083330437	0.102786	0.998873	0.998291
180	0.082191331	0.090256	0.999155	0.998291
181	0.084583756	0.095436	0.998873	0.998291
182	0.10323477	0.109677	0.99662	0.996581
183	0.089624305	0.104584	0.997887	0.997721
184	0.093377471	0.096466	0.998732	0.998291
185	0.088096082	0.104976	0.998732	0.997721
186	0.086322267	0.091855	0.999437	0.997721
187	0.093071417	0.09981	0.996901	0.998291
188	0.099019032	0.10407	0.998028	0.997721
189	0.091236188	0.106914	0.999014	0.998291
190	0.088449859	0.092932	0.999577	0.998291

191	0.091027145	0.109309	0.999155	0.998291
192	0.089013257	0.097151	0.998873	0.997151
193	0.085520167	0.100935	0.998592	0.997721
194	0.082086001	0.086652	0.999577	0.998291
195	0.07891793	0.089799	0.999296	0.998291
196	0.07823303	0.085838	0.999437	0.998291
197	0.077330326	0.090483	0.999014	0.998291
198	0.076629187	0.089798	0.998873	0.998291
199	0.075477714	0.083545	0.999155	0.998291
200	0.07447986	0.106705	0.995352	0.996581
201	0.074571213	0.081344	0.999718	0.998291
202	0.07343387	0.081346	0.999014	0.998291
203	0.07076664	0.083097	0.999437	0.998291
204	0.068518095	0.076151	0.999577	0.997721
205	0.069458033	0.075292	0.999155	0.998291
206	0.067833934	0.085171	0.997465	0.998291
207	0.070170819	0.094427	0.998732	0.997721
208	0.070942571	0.077008	0.998873	0.998291
209	0.068543579	0.077344	0.999437	0.998291
210	0.068137976	0.077732	0.999155	0.998291
211	0.065034394	0.076404	0.999718	0.998291
212	0.067330495	0.072363	0.999718	0.998291

213	0.067975347	0.075108	0.999014	0.997721
214	0.067707643	0.079387	0.999718	0.997721
215	0.066638189	0.074189	0.998592	0.998291
216	0.066750446	0.073733	0.999155	0.998291
217	0.064074986	0.07592	0.999718	0.997721
218	0.064790712	0.072244	0.998592	0.997721
219	0.066563705	0.076794	0.999014	0.998291
220	0.073468164	0.092012	0.996056	0.998291
221	0.07176942	0.085626	0.99831	0.997721
222	0.071574721	0.090306	0.997324	0.997721
223	0.079955722	0.09135	0.997887	0.998291
224	0.076105473	0.082828	0.99831	0.997721
225	0.078961236	0.096132	0.99831	0.998291
226	0.078505139	0.082494	0.999155	0.998291
227	0.075292927	0.085538	0.999718	0.998291
228	0.074108558	0.081221	0.999014	0.998291
229	0.072197296	0.07732	0.999859	0.998291
230	0.07093558	0.076319	0.998732	0.998291
231	0.068216658	0.079378	0.999577	0.998291
232	0.06651794	0.072418	1	0.998291
233	0.066003107	0.068336	1	0.998291
234	0.064765651	0.073366	0.998873	0.998291

235	0.066943178	0.075269	0.999437	0.998291
236	0.066840029	0.082486	0.997465	0.998291
237	0.070722706	0.09324	0.996761	0.997721
238	0.086074304	0.089324	0.999155	0.997721
239	0.077130188	0.093186	0.997183	0.998291
240	0.075277474	0.089424	0.998873	0.998291
241	0.07289649	0.082443	0.998028	0.998291
242	0.072244006	0.079996	0.998873	0.998291
243	0.072814996	0.076893	0.999296	0.998291
244	0.069209508	0.088457	0.999437	0.998291
245	0.075068254	0.081447	0.998592	0.997721
246	0.073802345	0.080765	0.997606	0.998291
247	0.081567836	0.099199	0.996338	0.998291
248	0.09140155	0.085902	0.998732	0.998291
249	0.087247955	0.100139	0.997324	0.998291
250	0.087847923	0.109095	0.996197	0.997721
251	0.088986748	0.099718	0.998732	0.998291
252	0.088283875	0.098241	0.998028	0.998291
253	0.086023831	0.096652	0.998732	0.997151
254	0.08275367	0.094566	0.997887	0.997721
255	0.083401011	0.088065	0.999296	0.997721
256	0.082030849	0.099437	0.999014	0.997151

257	0.077221944	0.085443	0.999577	0.998291
258	0.074944013	0.106415	0.998592	0.998291
259	0.07703655	0.086855	0.999577	0.998291
260	0.076100379	0.081142	0.999718	0.998291
261	0.073869741	0.092248	0.997746	0.998291
262	0.072174005	0.077151	0.999437	0.998291
263	0.070478197	0.080253	0.999155	0.998291
264	0.069751929	0.076278	0.999718	0.998291
265	0.067519228	0.072411	0.999577	0.998291
266	0.068650641	0.081964	0.998592	0.998291
267	0.069240974	0.079814	0.998028	0.998291
268	0.075049031	0.096269	0.996479	0.997151
269	0.073697016	0.083411	0.997465	0.998291
270	0.092846723	0.117673	0.993521	0.994872
271	0.09148244	0.093613	0.998873	0.998291
272	0.094970753	0.131318	0.997465	0.997151
273	0.091808569	0.107817	0.999155	0.997721
274	0.090027584	0.098041	0.999155	0.998291
275	0.084941388	0.089787	0.998451	0.998291
276	0.082681434	0.097717	0.999014	0.998291
277	0.078596065	0.083072	0.999859	0.998291
278	0.076867156	0.083684	0.999437	0.998291

279	0.076168928	0.088651	0.999577	0.997721
280	0.074451056	0.083424	0.999577	0.998291
281	0.0768792	0.082984	0.997746	0.998291
282	0.083342038	0.100587	0.997887	0.997721
283	0.082832647	0.090523	0.997887	0.997721
284	0.085790792	0.100384	0.999155	0.997721
285	0.085592401	0.092824	0.999014	0.997721
286	0.080756908	0.094637	0.998873	0.997721
287	0.080278624	0.084923	0.999437	0.998291
288	0.077594671	0.095038	0.999155	0.998291
289	0.08300124	0.09676	0.998592	0.996581
290	0.078970228	0.087783	0.998592	0.998291
291	0.076855761	0.083549	0.999437	0.998291
292	0.073500355	0.079954	1	0.998291
293	0.071376267	0.077282	0.999577	0.998291
294	0.077592882	0.08529	0.999437	0.998291
295	0.0746906	0.08167	0.999014	0.998291
296	0.073073977	0.083583	0.999296	0.998291
297	0.072745746	0.079044	1	0.998291
298	0.06957734	0.075023	0.999437	0.998291
299	0.068791735	0.081333	0.999577	0.997151
300	0.067633089	0.080532	0.999859	0.998291

301	0.068551222	0.074618	0.999718	0.997721
302	0.068039486	0.073137	0.999859	0.998291
303	0.076399864	0.091786	0.996197	0.997721
304	0.075486426	0.08989	0.99831	0.998291
305	0.076699755	0.093123	0.998873	0.998291
306	0.075144218	0.080945	0.999859	0.998291
307	0.074328217	0.080127	0.999437	0.998291
308	0.07699222	0.088362	0.999296	0.998291
309	0.076933961	0.080411	0.999437	0.997721
310	0.073401457	0.083268	0.999718	0.998291
311	0.07282249	0.07916	0.998592	0.998291
312	0.071371622	0.086274	0.999296	0.998291
313	0.074887848	0.126751	0.992113	0.996581
314	0.075642924	0.087282	0.997606	0.998291
315	0.076425847	0.085895	0.999155	0.998291
316	0.073311983	0.087654	0.999296	0.997721
317	0.07424791	0.07717	0.999014	0.998291
318	0.083866941	0.134864	0.992113	0.982906
319	0.092542987	0.103944	0.997465	0.997721
320	0.089339382	0.119875	0.997042	0.998291
321	0.092028715	0.105734	0.99831	0.998291
322	0.08747408	0.099593	0.999014	0.998291

323	0.088777444	0.10093	0.997746	0.998291
324	0.088771241	0.111281	0.998873	0.998291
325	0.092688254	0.098953	0.998592	0.998291
326	0.088563183	0.103604	0.999718	0.998291
327	0.087082756	0.101171	0.996901	0.998291
328	0.084766395	0.091997	0.999014	0.998291
329	0.083126828	0.096631	0.998451	0.997721
330	0.079972359	0.093093	0.999577	0.998291
331	0.078484214	0.092402	0.99831	0.997721
332	0.076923238	0.082808	0.999155	0.998291
333	0.074472225	0.082551	1	0.998291
334	0.072021626	0.077522	0.999718	0.998291
335	0.072826476	0.0845	0.997887	0.997151
336	0.088250805	0.110951	0.997042	0.997721
337	0.086558928	0.096332	0.999437	0.998291
338	0.078065781	0.090474	0.998732	0.998291
339	0.080614428	0.086623	0.999437	0.998291
340	0.077774033	0.091075	0.999437	0.998291
341	0.089406069	0.097326	0.997746	0.997721
342	0.084345886	0.091793	0.998451	0.998291
343	0.084568884	0.093909	0.99831	0.998291
344	0.086662417	0.099256	0.997465	0.998291

345	0.081669918	0.098931	0.998592	0.998291
346	0.083982435	0.092883	0.999155	0.998291
347	0.079623155	0.091148	0.999437	0.998291
348	0.078367965	0.08596	0.999014	0.998291
349	0.079004489	0.092538	0.998732	0.998291
350	0.074933302	0.085502	0.999155	0.998291
351	0.074456759	0.080869	0.999437	0.998291
352	0.073585661	0.081056	0.999014	0.998291
353	0.090598145	0.105358	0.997042	0.998291
354	0.089775086	0.099162	0.997606	0.998291
355	0.092807027	0.102171	0.99493	0.997721
356	0.089422639	0.101445	0.999014	0.998291
357	0.087527164	0.091797	0.998873	0.997151
358	0.097168409	0.101935	0.997746	0.998291
359	0.089046528	0.096108	0.999155	0.998291
360	0.088486769	0.095941	0.999014	0.998291
361	0.085036843	0.103739	0.999014	0.998291
362	0.086661917	0.089452	0.999155	0.998291
363	0.083052373	0.089958	0.999155	0.998291
364	0.081776598	0.08882	0.999437	0.998291
365	0.079112457	0.085527	0.999718	0.997721
366	0.077540762	0.085893	0.999577	0.998291

367	0.086555784	0.097633	0.995634	0.997721
368	0.089125432	0.089265	0.997465	0.997721
369	0.081081331	0.094408	0.998873	0.998291
370	0.083347753	0.099258	0.997606	0.998291
371	0.084304618	0.094526	0.998451	0.997721
372	0.079601833	0.090632	0.998732	0.998291
373	0.079112366	0.084555	0.999859	0.998291
374	0.078565121	0.085148	1	0.998291
375	0.074701271	0.081265	0.999296	0.998291
376	0.074073213	0.082128	0.999577	0.998291
377	0.073994158	0.07873	0.999437	0.998291
378	0.071391935	0.078496	0.999718	0.998291
379	0.071482899	0.076107	1	0.998291
380	0.072296049	0.100154	0.995352	0.994872
381	0.069768117	0.085056	0.997887	0.997721
382	0.070175775	0.082879	0.999014	0.998291
383	0.070302563	0.080544	0.999577	0.998291
384	0.068231202	0.082086	0.999859	0.998291
385	0.06913286	0.077258	0.999718	0.998291
386	0.068342101	0.082699	0.999437	0.998291
387	0.065336354	0.074054	0.999718	0.998291
388	0.064268089	0.096728	0.995775	0.997151

389	0.063772301	0.0742	0.998873	0.998291
390	0.064001974	0.068939	0.999577	0.998291
391	0.063292852	0.09576	0.994789	0.998291
392	0.068586596	0.073977	0.998732	0.998291
393	0.069446565	0.109469	0.992958	0.996011
394	0.077054382	0.10394	0.997465	0.998291
395	0.07606944	0.098913	0.998732	0.998291
396	0.074330578	0.089974	0.999014	0.998291
397	0.072142844	0.08244	0.999437	0.998291
398	0.080048729	0.094771	0.999155	0.998291
399	0.08335042	0.089659	0.999718	0.998291
400	0.085961166	0.09456	0.99831	0.998291
401	0.085264787	0.091697	0.998873	0.998291
402	0.08039767	0.085945	0.999859	0.998291
403	0.078038075	0.087344	0.999155	0.998291
404	0.077635603	0.085927	0.999718	0.998291
405	0.079466639	0.088791	0.999577	0.998291
406	0.077377613	0.089963	0.998732	0.998291
407	0.077678252	0.082164	0.999718	0.998291
408	0.075260542	0.100259	0.997746	0.998291
409	0.080320445	0.091112	0.998873	0.998291
410	0.087706223	0.094468	0.997746	0.998291

411	0.082846221	0.087577	0.998732	0.998291
412	0.078473993	0.086788	0.999296	0.998291
413	0.080075107	0.090764	0.999296	0.998291
414	0.07621449	0.086621	0.998873	0.998291
415	0.076532041	0.086896	0.999437	0.998291
416	0.072551509	0.080041	0.999577	0.998291
417	0.074159769	0.079522	0.999296	0.998291
418	0.078104668	0.077374	0.999577	0.998291
419	0.073639216	0.082112	0.999014	0.998291
420	0.075329311	0.080299	0.999577	0.997721
421	0.070611915	0.081777	0.998169	0.998291
422	0.070054706	0.07554	1	0.998291
423	0.068605568	0.084455	0.999437	0.998291
424	0.066867249	0.087648	0.998732	0.998291
425	0.066386909	0.07035	0.999859	0.998291
426	0.068025302	0.075802	0.999014	0.998291
427	0.071101577	0.077829	0.997887	0.998291
428	0.068827657	0.078934	0.998592	0.997721
429	0.069950966	0.075735	0.99831	0.998291
430	0.075869353	0.080808	0.997324	0.997151
431	0.075369237	0.077802	0.999014	0.998291
432	0.074952577	0.087478	0.998592	0.998291

433	0.075388784	0.082934	0.999437	0.997721
434	0.07732414	0.108077	0.996479	0.982906
435	0.08821587	0.10033	0.994648	0.997721
436	0.083792241	0.106837	0.996056	0.998291
437	0.092376282	0.105223	0.998169	0.998291
438	0.087374765	0.10508	0.999296	0.998291
439	0.083944762	0.096784	0.998732	0.998291
440	0.087414799	0.101377	0.999718	0.998291
441	0.081810053	0.086273	0.999859	0.998291
442	0.079378961	0.08316	0.999718	0.998291
443	0.076848528	0.081562	0.999718	0.998291
444	0.076236893	0.082849	0.999296	0.998291
445	0.078946434	0.08721	0.998732	0.998291
446	0.074352394	0.079882	0.999718	0.998291
447	0.073117554	0.083236	1	0.998291
448	0.08527474	0.087171	0.998451	0.997721
449	0.083694746	0.119609	0.991831	0.996581
450	0.083839961	0.093084	0.998592	0.998291
451	0.096079425	0.110255	0.997042	0.998291
452	0.089271456	0.105342	0.998732	0.998291
453	0.090916554	0.106245	0.997887	0.997151
454	0.087605213	0.104782	0.999155	0.998291

455	0.08955026	0.104215	0.998028	0.997151
456	0.088622601	0.097405	0.999014	0.998291
457	0.085356678	0.108558	0.999718	0.998291
458	0.087252303	0.091045	0.999718	0.998291
459	0.083631874	0.093047	0.999014	0.997721
460	0.082978905	0.087327	0.999859	0.998291
461	0.079602801	0.087381	0.999577	0.998291
462	0.083948787	0.104136	0.995916	0.997151
463	0.081716278	0.086287	0.998732	0.998291
464	0.083392252	0.103159	0.999014	0.998291
465	0.085228103	0.092677	0.998451	0.998291
466	0.083458551	0.088499	0.999014	0.998291
467	0.082297135	0.089711	0.998592	0.998291
468	0.094772497	0.102289	0.998592	0.997721
469	0.08647224	0.09419	0.998873	0.998291
470	0.082684972	0.091367	0.999718	0.998291
471	0.084276596	0.097677	0.999718	0.998291
472	0.085452001	0.09253	0.998873	0.998291
473	0.087896605	0.099982	0.998592	0.998291
474	0.085615454	0.091106	0.999296	0.998291
475	0.087280923	0.091606	0.998732	0.998291
476	0.095429565	0.100651	0.998028	0.997721

477	0.08933591	0.097284	0.998451	0.998291
478	0.089617404	0.094912	0.998169	0.998291
479	0.088868804	0.089484	0.998732	0.998291
480	0.088637009	0.09196	0.999014	0.998291
481	0.08517016	0.090032	0.999577	0.998291
482	0.089001819	0.091327	0.999577	0.998291
483	0.085208594	0.099085	0.999155	0.998291
484	0.085188028	0.087782	0.999296	0.998291
485	0.081103075	0.091132	1	0.998291
486	0.081126496	0.087586	0.999296	0.998291
487	0.079018945	0.086959	0.999718	0.998291
488	0.078710312	0.088814	0.999014	0.998291
489	0.077518263	0.083004	1	0.998291
490	0.073805267	0.083242	1	0.998291
491	0.072263574	0.082733	0.999718	0.998291
492	0.069989992	0.076652	0.999859	0.998291
493	0.067963138	0.073446	0.999859	0.998291
494	0.067035542	0.071816	0.999718	0.998291
495	0.06625443	0.071067	1	0.998291
496	0.065665172	0.069171	0.999718	0.998291
497	0.065612988	0.069948	0.999718	0.998291
498	0.065835802	0.069777	0.999718	0.998291

499	0.068042595	0.074381	0.999296	0.998291
500	0.073021692	0.089022	0.997746	0.997721
501	0.074288531	0.083992	0.999014	0.998291
502	0.082740396	0.089164	0.997465	0.998291
503	0.082204775	0.095773	0.996479	0.997721
504	0.086022131	0.099001	0.998028	0.998291
505	0.083061742	0.093403	0.999437	0.998291
506	0.082407392	0.101043	0.997887	0.997721
507	0.078908931	0.085275	0.999014	0.998291
508	0.07572506	0.083212	1	0.998291
509	0.073803193	0.080998	0.999859	0.998291
510	0.076394818	0.078578	0.999718	0.998291
511	0.07137094	0.076781	0.999577	0.998291
512	0.070438991	0.077149	0.999859	0.998291
513	0.069371752	0.07687	0.999296	0.998291
514	0.069073951	0.075213	1	0.998291
515	0.067546376	0.072444	0.999718	0.998291
516	0.067592276	0.085723	0.999437	0.998291
517	0.068334934	0.079468	0.998451	0.998291
518	0.076819078	0.091196	0.99831	0.997721
519	0.073794697	0.089269	0.998592	0.997721
520	0.073551839	0.090338	0.997606	0.997721

521	0.072408954	0.089862	0.998451	0.998291
522	0.072293475	0.079901	0.999014	0.998291
523	0.070185788	0.076621	0.999577	0.998291
524	0.070129425	0.075693	0.999859	0.998291
525	0.072679236	0.08649	0.999437	0.998291
526	0.070594649	0.082011	0.999155	0.998291
527	0.070478297	0.078095	0.999155	0.998291
528	0.071167333	0.077585	0.999437	0.998291
529	0.073122091	0.094353	0.999437	0.998291
530	0.070252969	0.07919	0.999437	0.998291
531	0.069293847	0.076283	0.999718	0.998291
532	0.068852515	0.076018	0.999296	0.997721
533	0.069903851	0.075148	0.999296	0.998291
534	0.072138005	0.078947	0.999296	0.998291
535	0.075289807	0.08501	0.999014	0.998291
536	0.069654317	0.08049	0.999296	0.998291
537	0.072430693	0.090006	0.999014	0.997721
538	0.072411482	0.077864	0.998732	0.998291
539	0.076127986	0.096643	0.998732	0.996581
540	0.073151056	0.081763	0.997324	0.998291
541	0.073740005	0.084029	0.999014	0.998291
542	0.077592687	0.091692	0.998451	0.998291

543	0.075892183	0.091426	0.99831	0.998291
544	0.09463093	0.093883	0.998028	0.998291
545	0.088306982	0.090646	0.998451	0.998291
546	0.084065416	0.098616	0.998873	0.997721
547	0.083206129	0.094877	0.997606	0.997721
548	0.079952137	0.091343	0.999155	0.998291
549	0.080923394	0.094432	0.997042	0.998291
550	0.086295681	0.094261	0.999014	0.998291
551	0.08580576	0.09041	0.998732	0.998291
552	0.084561535	0.095451	0.999437	0.998291
553	0.083960947	0.092304	0.999296	0.998291
554	0.083884616	0.087377	0.999718	0.998291
555	0.080253681	0.090835	0.998732	0.998291
556	0.078375984	0.088223	0.999155	0.998291
557	0.075451025	0.082292	0.999859	0.998291
558	0.078094074	0.078957	0.999577	0.998291
559	0.075412339	0.081902	0.999718	0.998291
560	0.074406705	0.079846	0.999437	0.998291
561	0.072845939	0.078278	0.999014	0.998291
562	0.071289349	0.076534	0.999155	0.998291
563	0.069419184	0.075426	0.999718	0.998291
564	0.068787196	0.088592	0.999296	0.998291

565	0.074509737	0.08093	0.999296	0.998291
566	0.069595683	0.082172	1	0.998291
567	0.067841559	0.074904	0.999859	0.998291
568	0.067653481	0.074617	0.999718	0.998291
569	0.072555999	0.089428	0.997746	0.998291
570	0.075850485	0.081807	0.999437	0.998291
571	0.071585602	0.081241	0.999577	0.998291
572	0.072482462	0.083523	0.999014	0.998291
573	0.072121406	0.083984	0.999155	0.998291
574	0.070977293	0.085143	0.999718	0.998291
575	0.07464207	0.079551	0.999437	0.997721
576	0.073981408	0.081597	0.999014	0.998291
577	0.072253437	0.082774	0.998028	0.998291
578	0.076008364	0.090667	0.998873	0.998291
579	0.074754869	0.083298	0.997887	0.997721
580	0.075452698	0.085534	0.999718	0.998291
581	0.071623869	0.07989	0.999577	0.998291
582	0.077450679	0.085756	0.99831	0.998291
583	0.081252283	0.086887	0.998169	0.998291
584	0.078533971	0.091108	0.999577	0.998291
585	0.080389265	0.091111	0.999014	0.998291
586	0.078516426	0.091566	0.999718	0.998291

587	0.076018462	0.08181	0.999155	0.998291
588	0.077428332	0.089462	0.998451	0.997721
589	0.074959937	0.080321	0.998873	0.998291
590	0.072098812	0.078116	0.999577	0.998291
591	0.07103283	0.076799	0.999859	0.998291
592	0.0703958	0.076337	0.999718	0.998291
593	0.069023945	0.090755	0.997324	0.996581
594	0.067657176	0.073554	1	0.998291
595	0.06525018	0.071993	1	0.998291
596	0.066223342	0.071985	0.999718	0.998291
597	0.066660618	0.08482	0.999014	0.998291
598	0.065582235	0.076749	0.999437	0.998291
599	0.064609947	0.076233	0.999718	0.997721
600	0.066039903	0.071473	0.999296	0.998291

Appendix B

Appendix B presents the intermediate confusion matrices for using VGGNet on the LISA dataset.

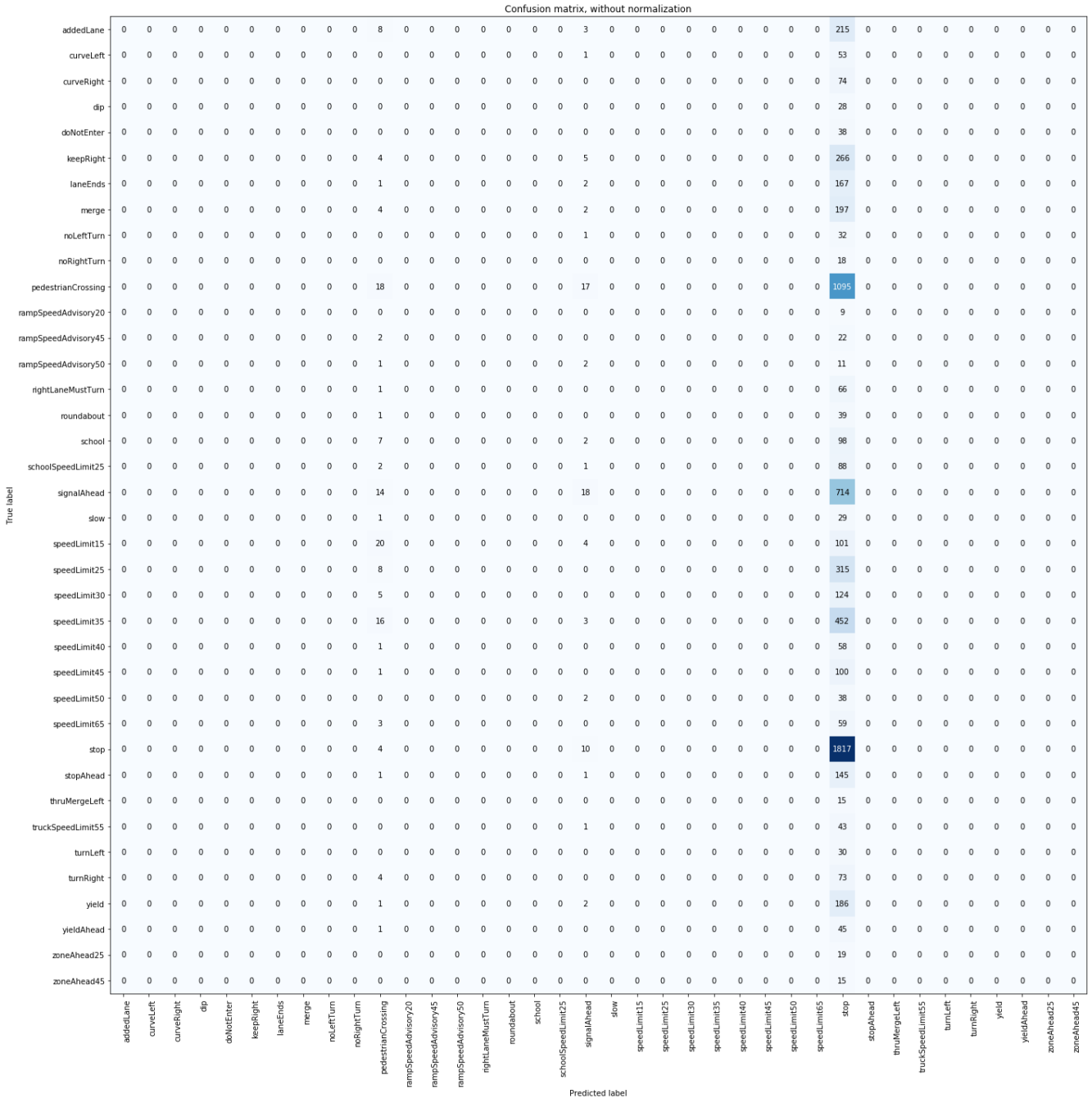

Figure B.1. Epoch 1 Confusion Matrix for VGGNet Model on LISA Dataset

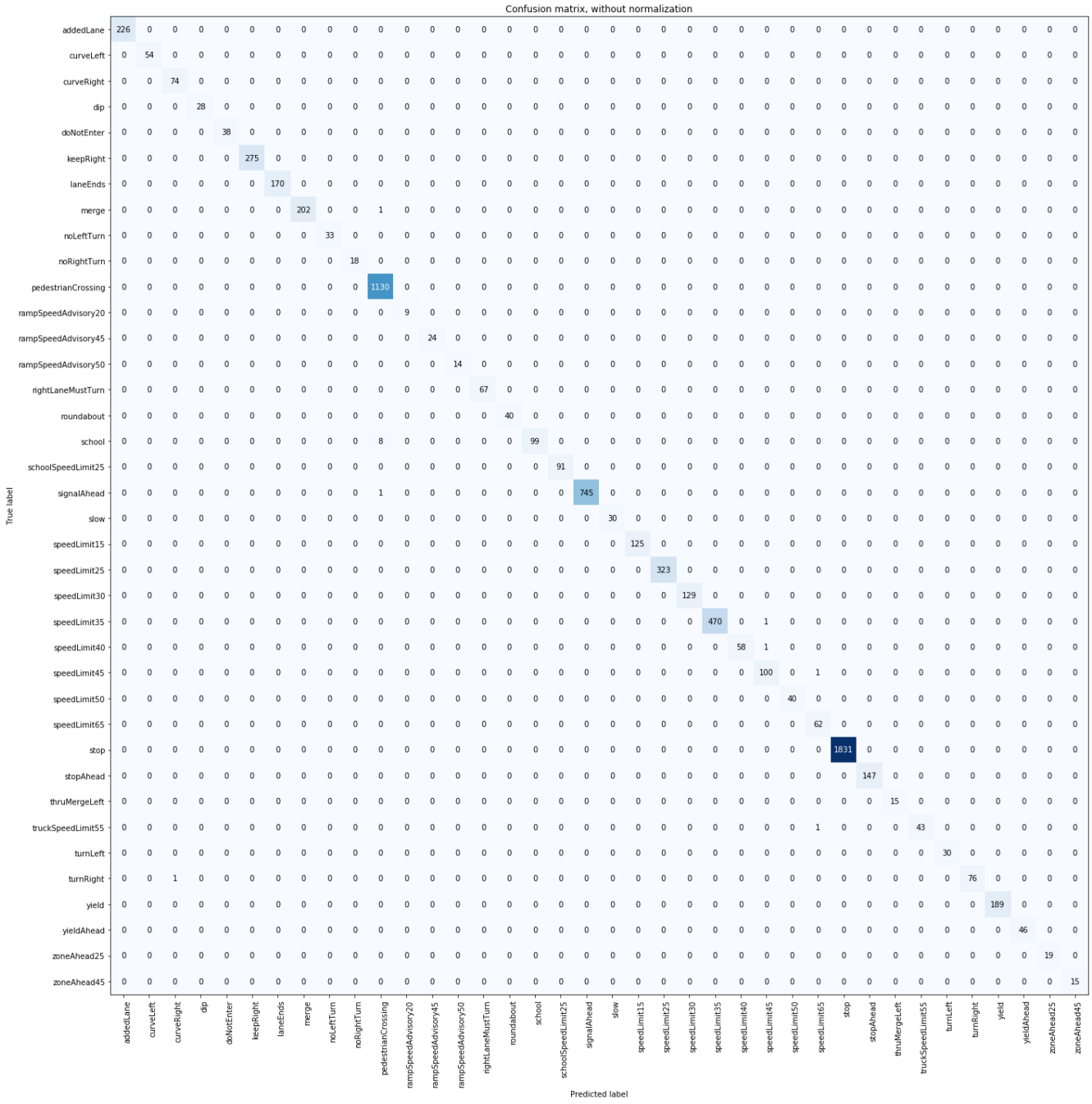

Figure B.2. Epoch 100 Confusion Matrix for VGGNet Model on LISA Dataset

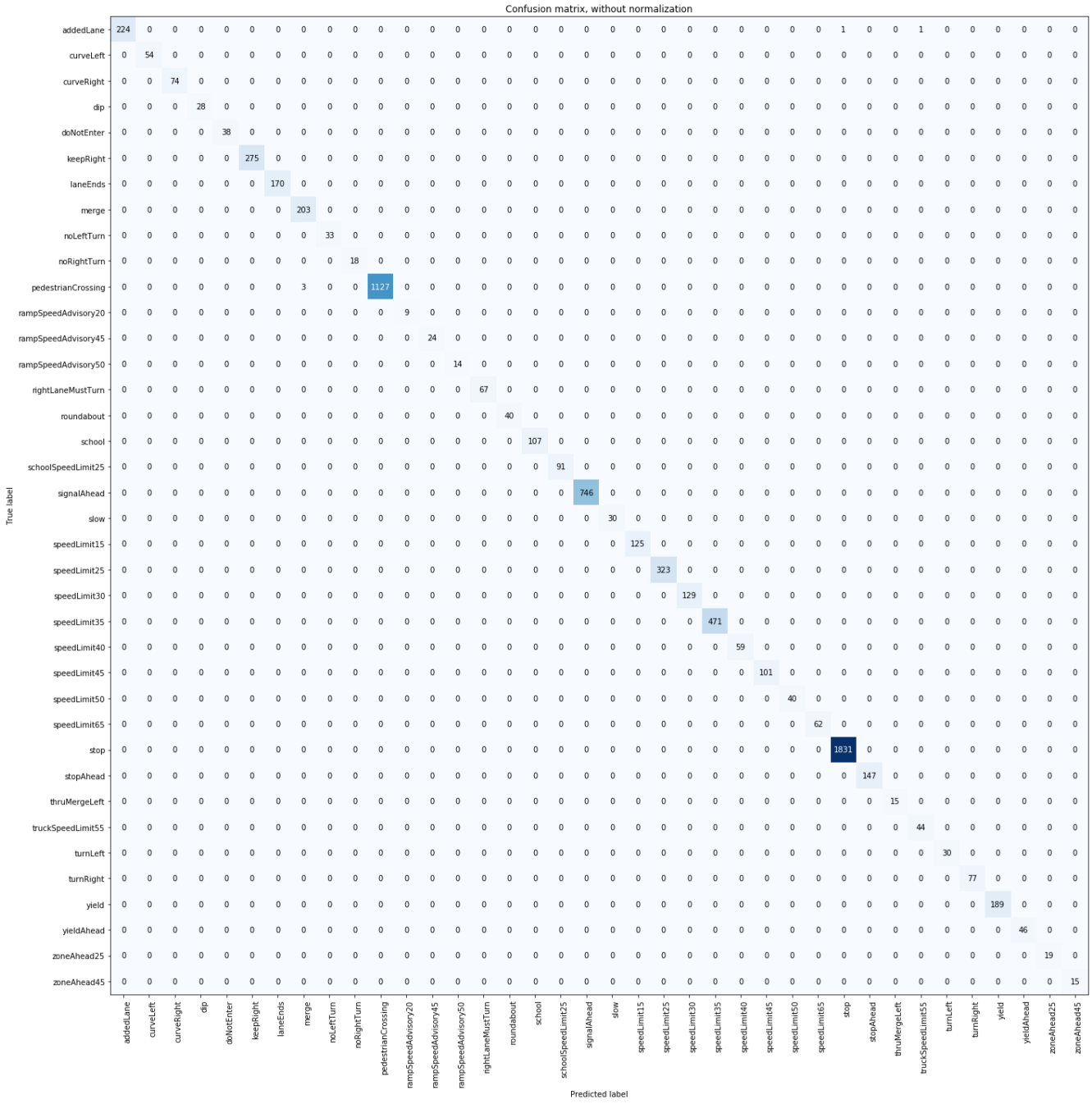

Figure B.3. Epoch 200 Confusion Matrix for VGGNet Model on LISA Dataset

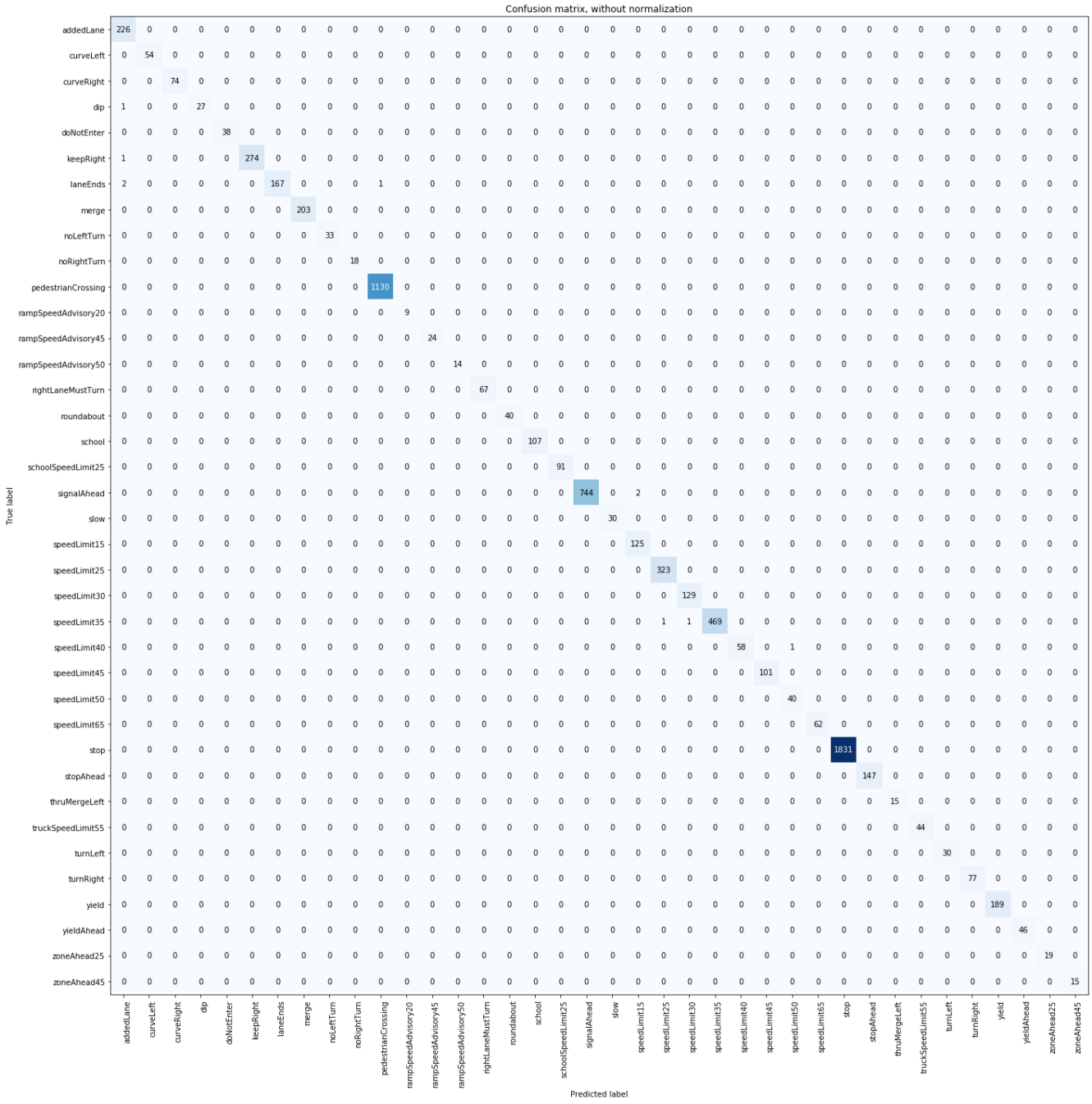

Figure B.4. Epoch 300 Confusion Matrix for VGGNet Model on LISA Dataset

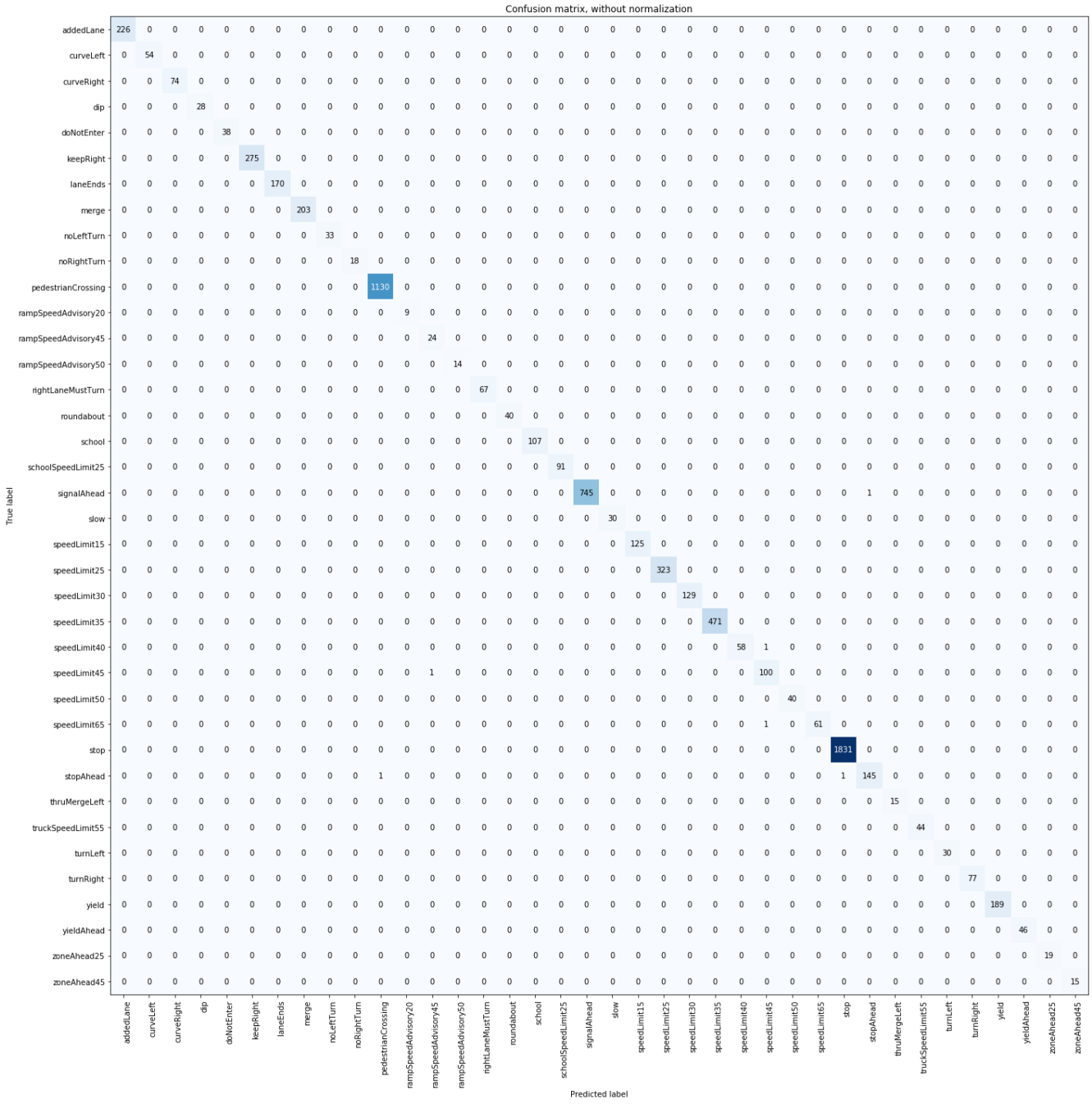

Figure B.5. Epoch 400 Confusion Matrix for VGGNet Model on LISA Dataset

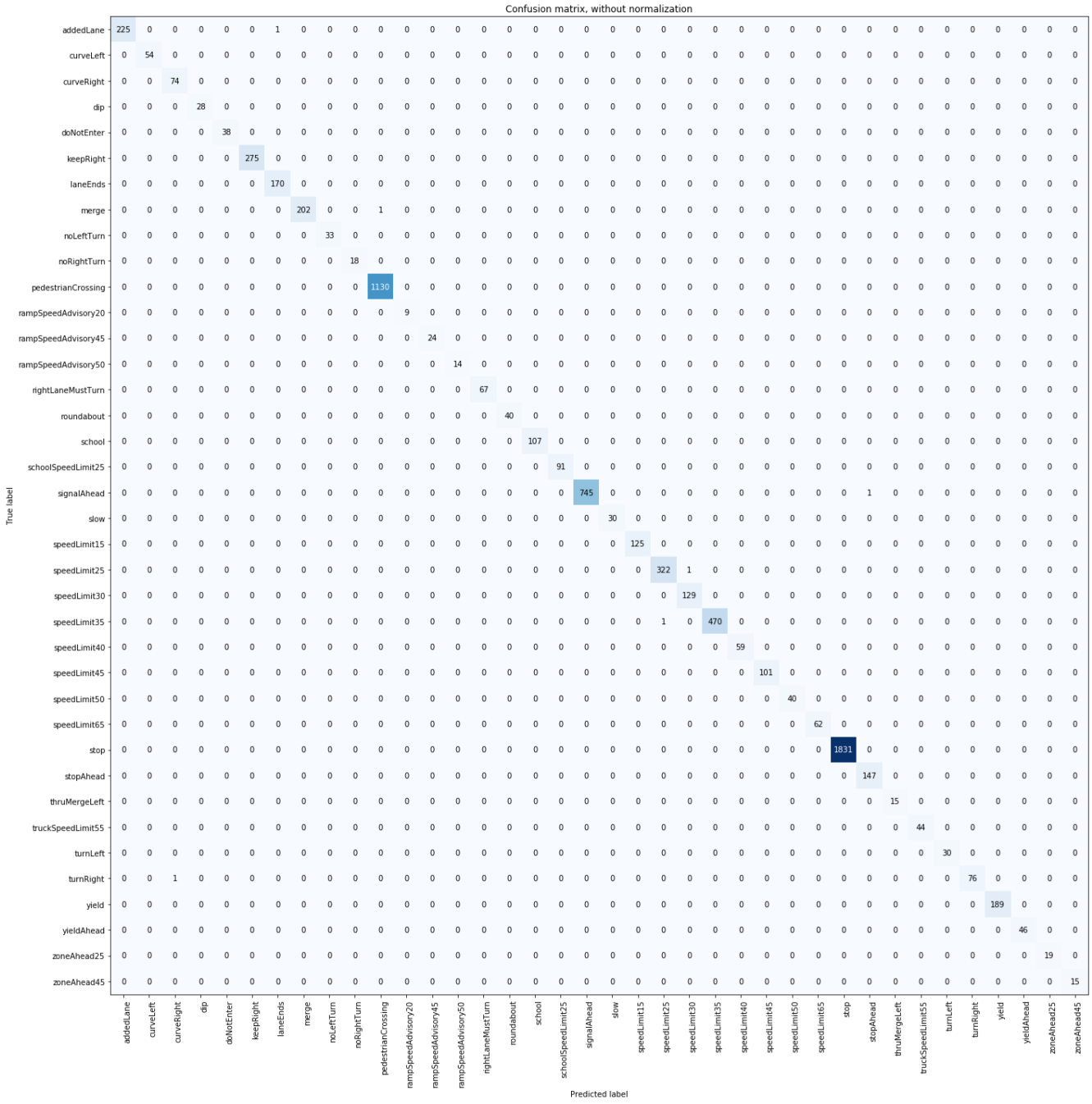

Figure B.6. Epoch 500 Confusion Matrix for VGGNet Model on LISA Dataset

Appendix C

Appendix C presents intermediate confusion matrices for training NuNet on the LISA dataset.

The 1st, 100th, 200th, 300th, 400th and 500th epochs are presented in this appendix.

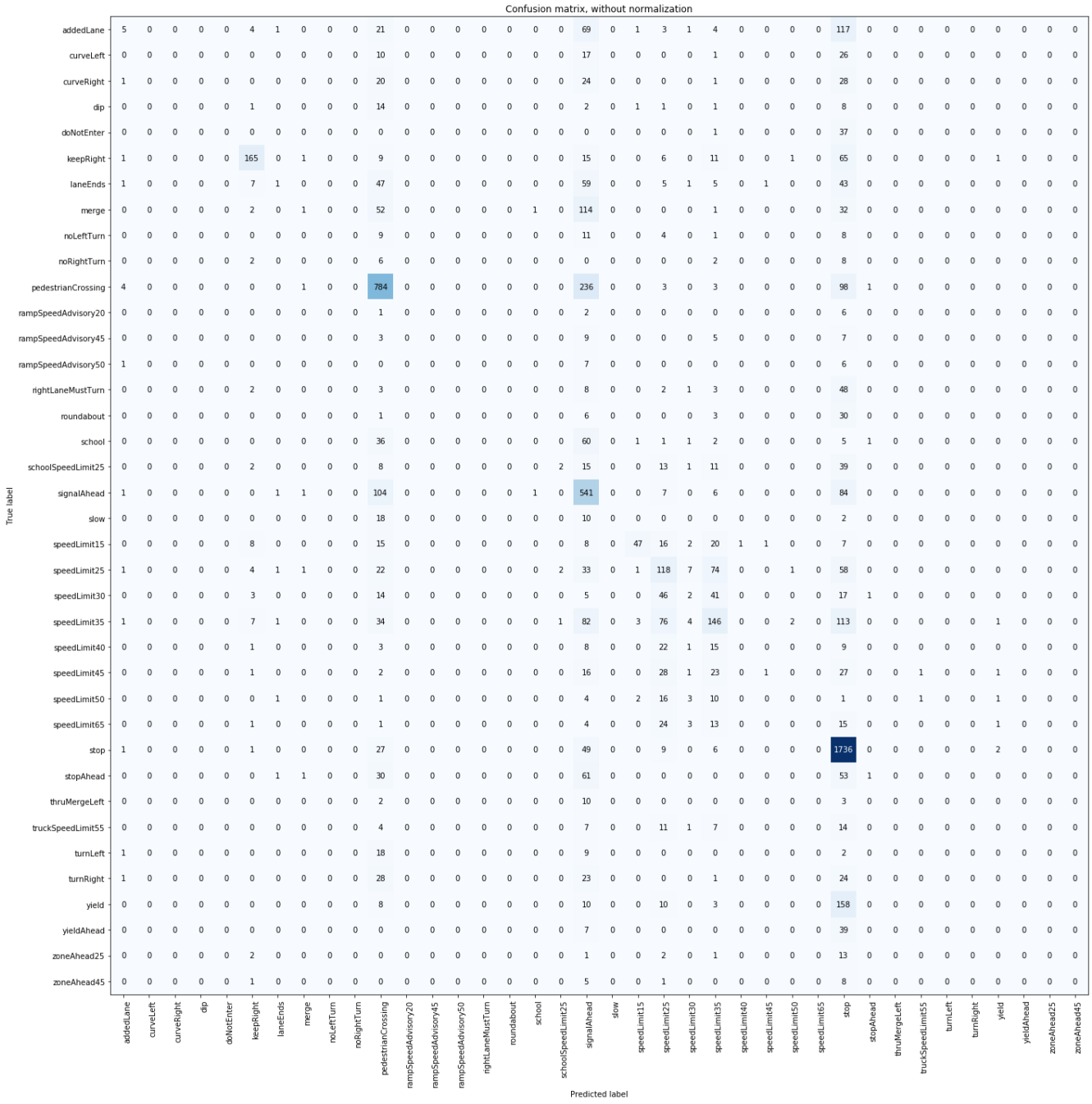

Figure C.7. Epoch 1 Confusion Matrix for NuNet Model on LISA Dataset

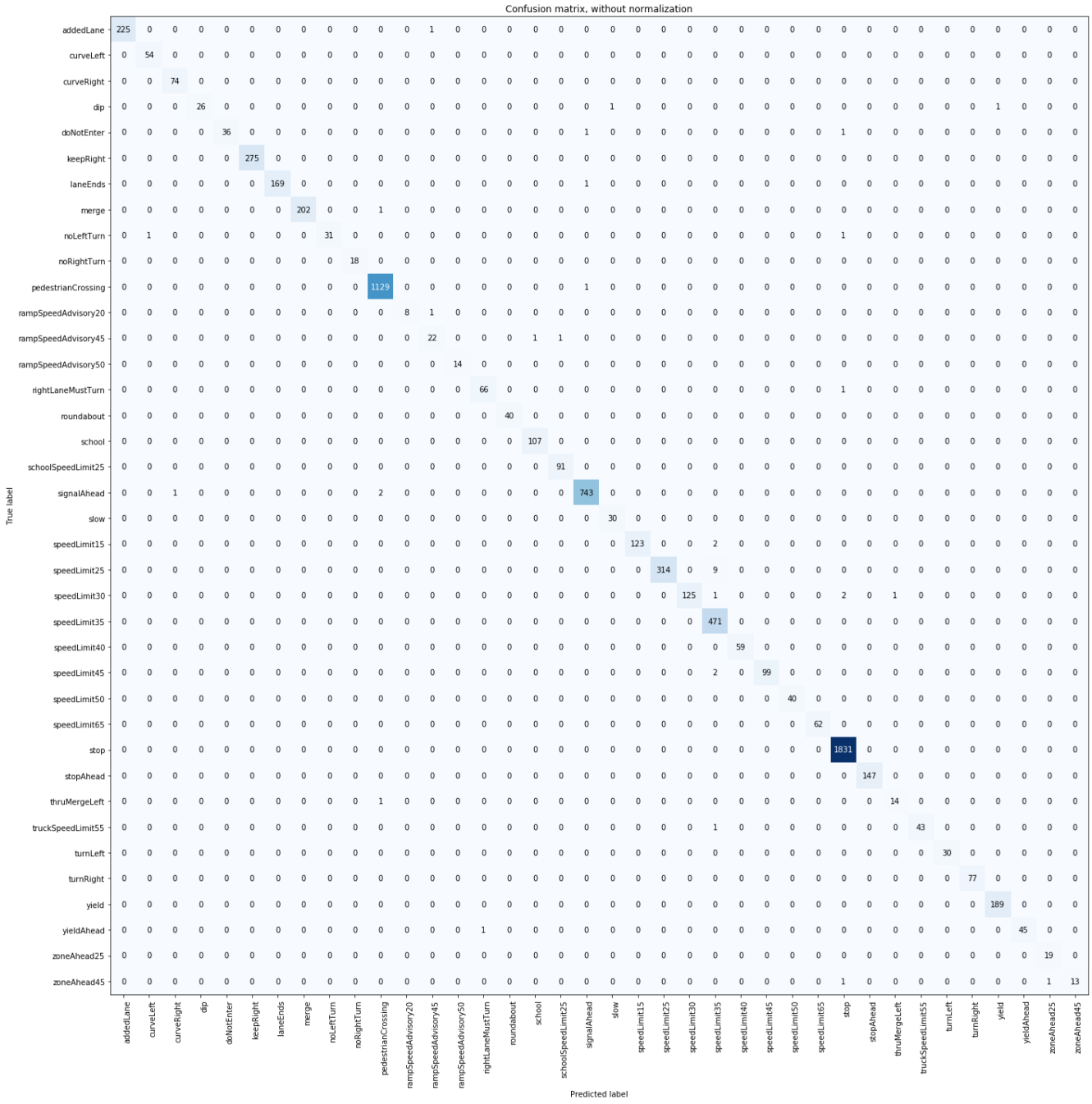

Figure C.8. Epoch 100 Confusion Matrix for NuNet Model on LISA Dataset

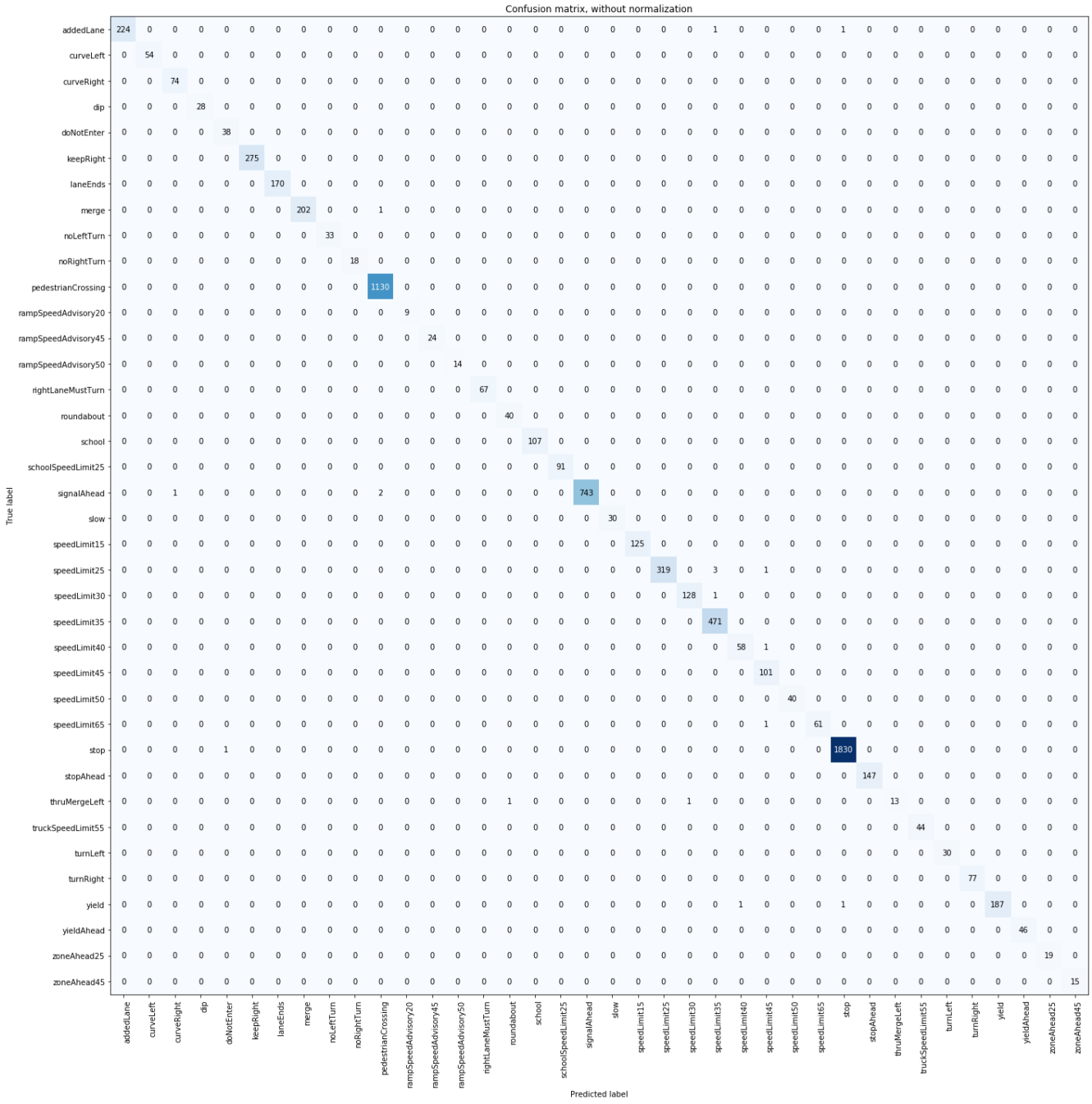

Figure C.9. Epoch 200 Confusion Matrix for NuNet Model on LISA Dataset

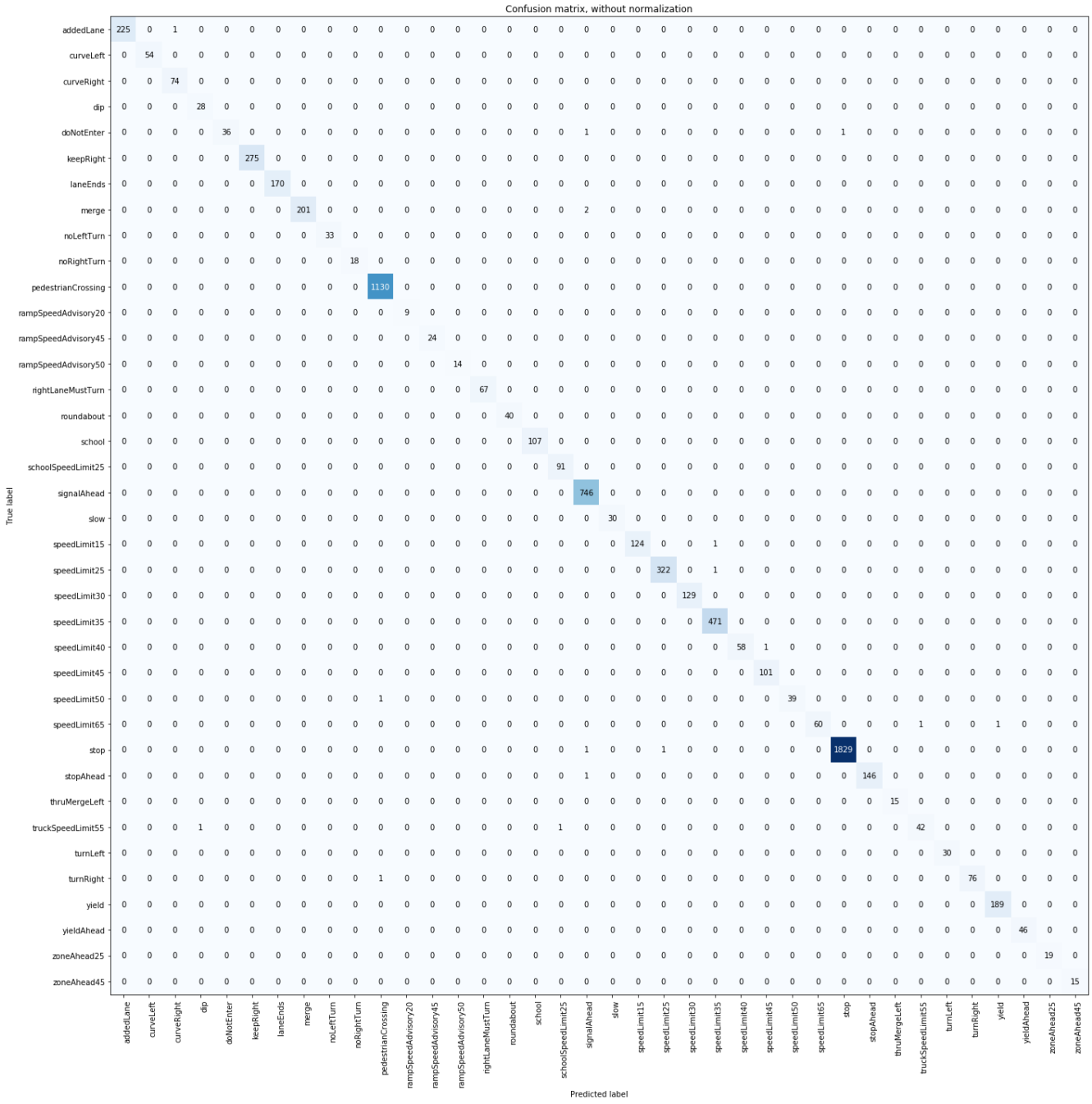

*Figure C.*10. Epoch 300 Confusion Matrix for NuNet Model on LISA Dataset

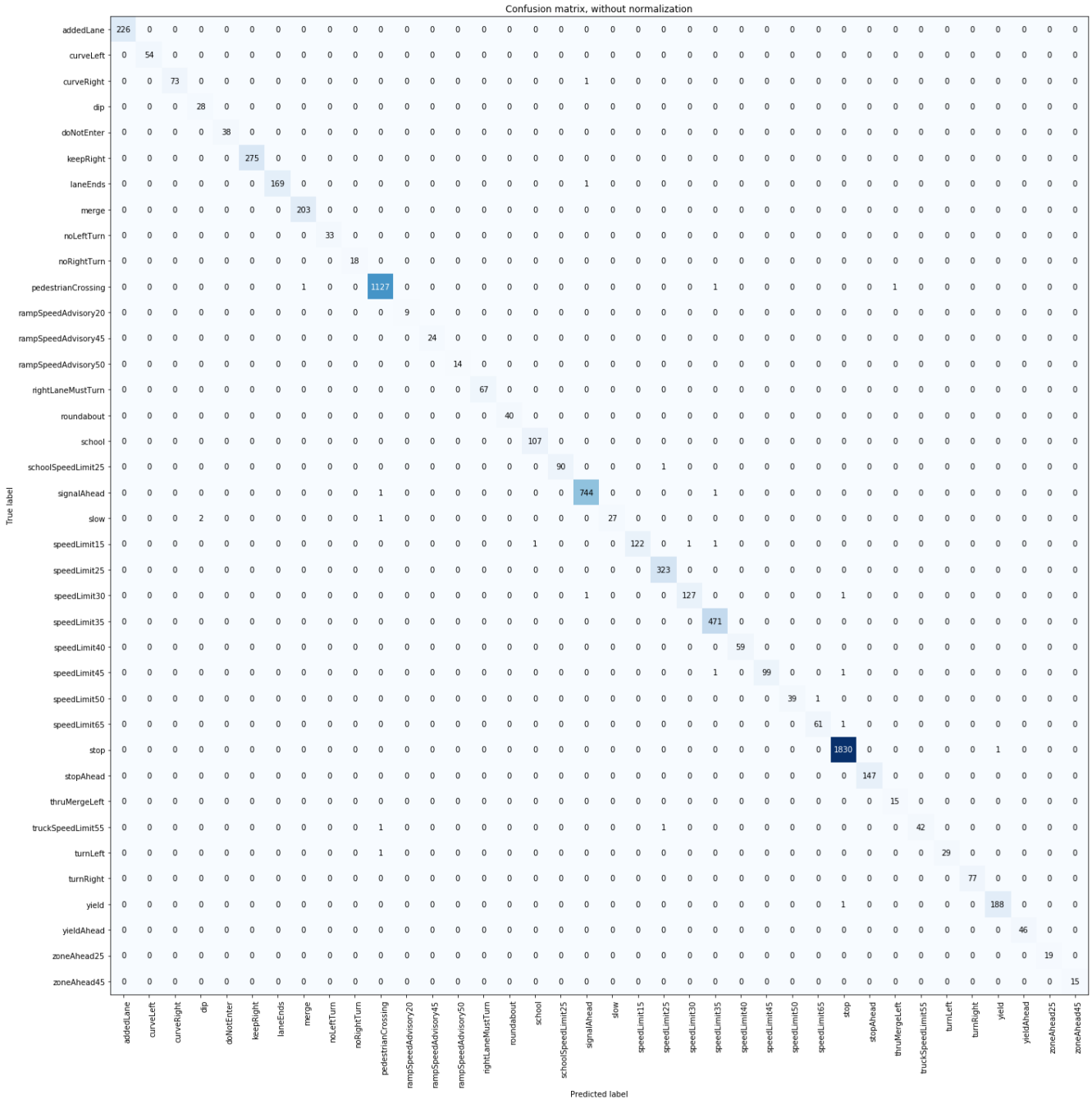

Figure C.11. Epoch 400 Confusion Matrix for NuNet Model on LISA Dataset

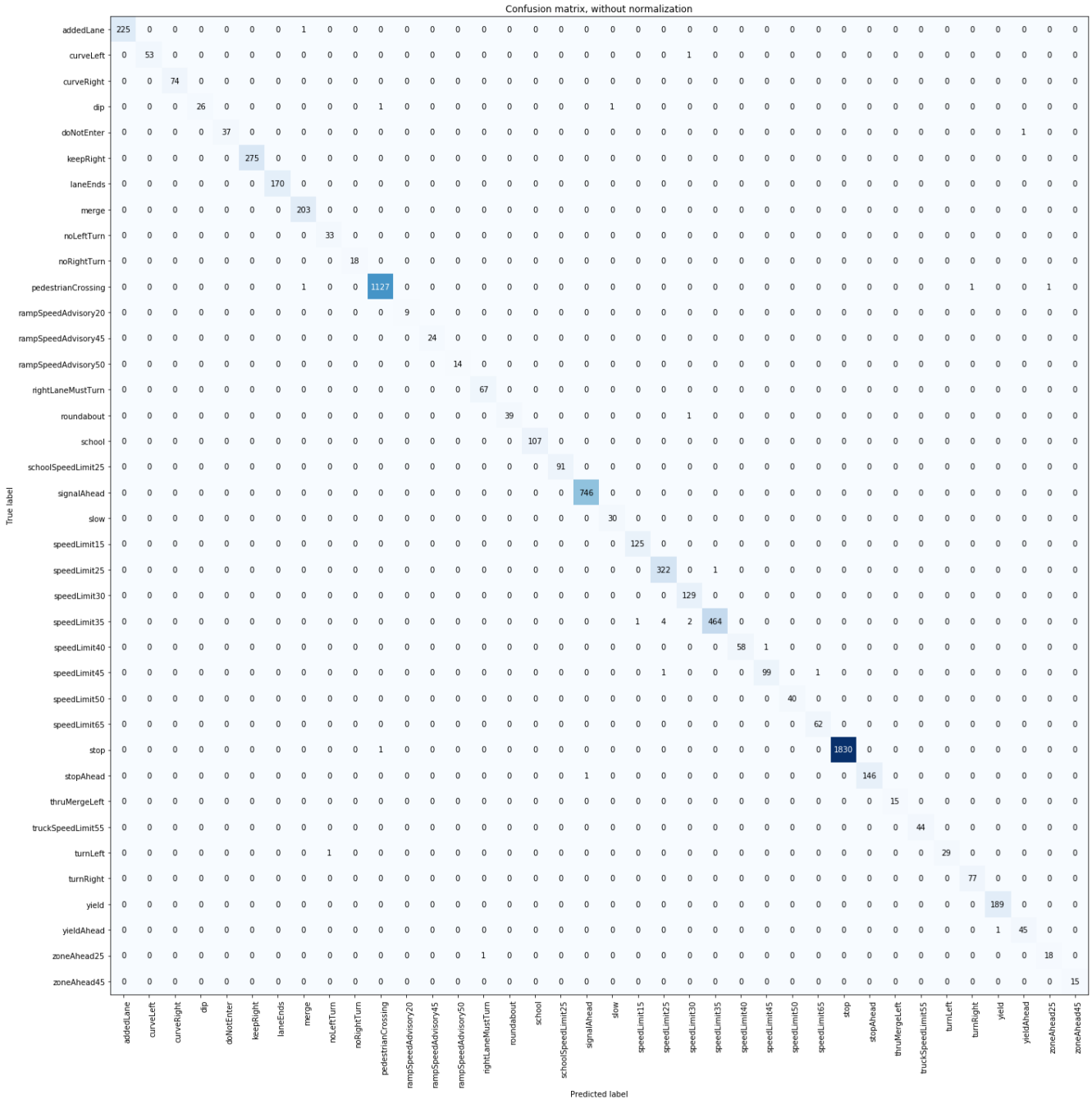

Figure C.12. Epoch 500 Confusion Matrix for NuNet Model on LISA Dataset

Appendix D

Appendix D presents intermediate confusion matrices for training NuNet with CIB dataset at 80/20 split for training and validation sets respectively. Confusion matrices of the 1st and 100th epochs are presented. The 200th, 300th, 400th and 500th epochs shared the same configuration as the 100th.

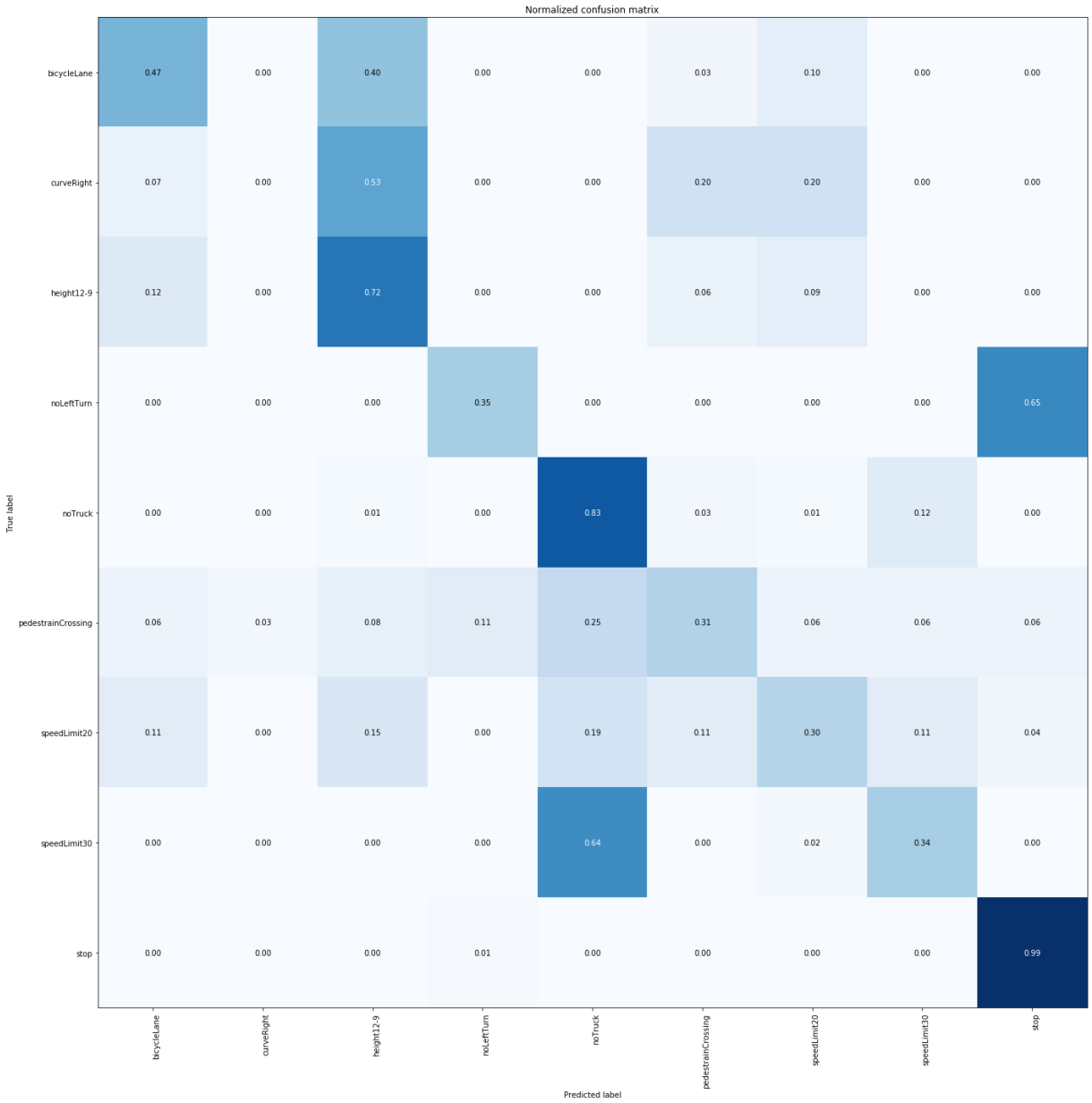

Figure D.13. Epoch 1 Confusion Matrix for NuNet Model on CIB Dataset

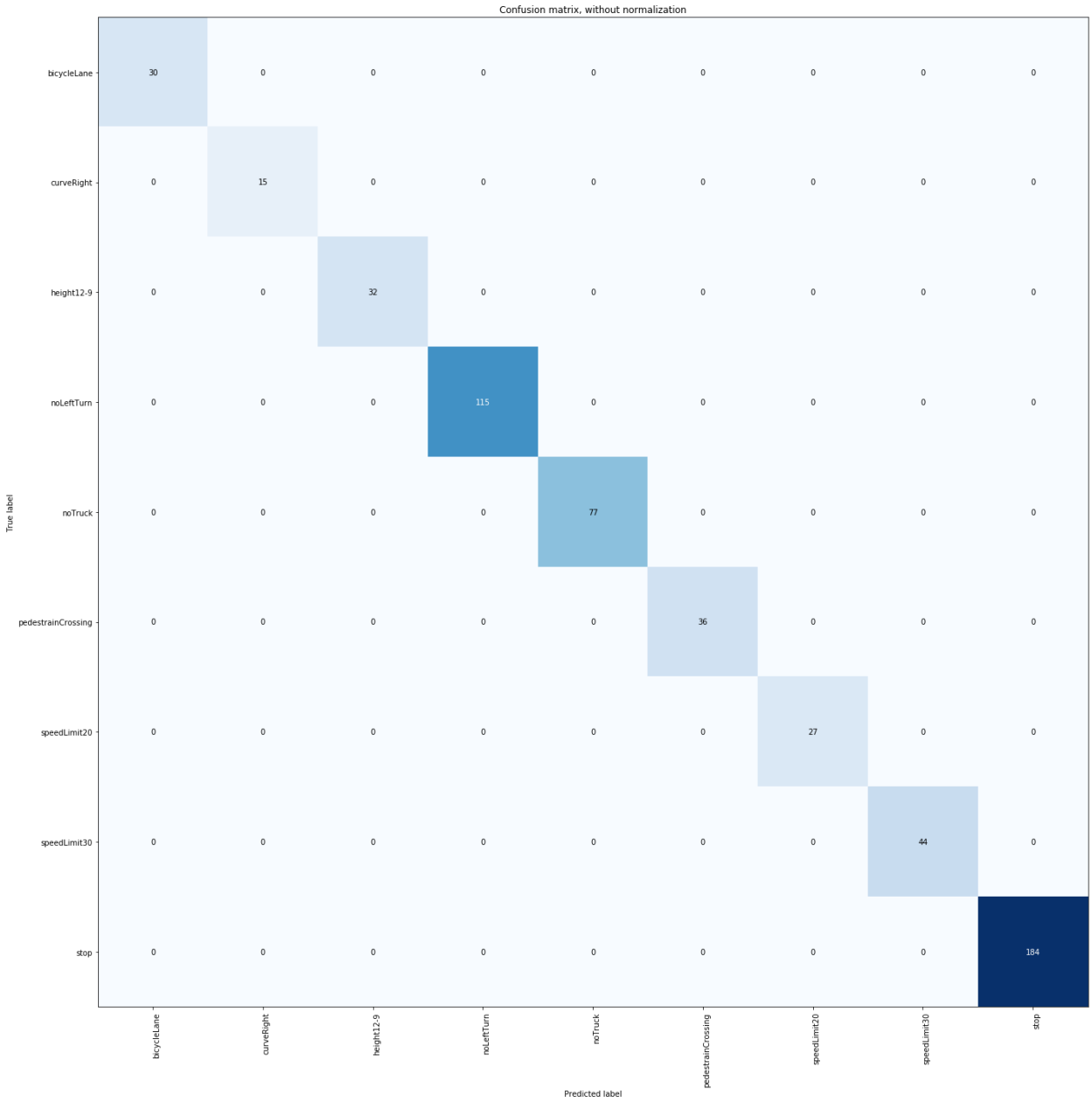

Figure D.14. Confusion Matrix for NuNet Model on CIB Dataset for 100th