Deep Learning, Neural Networks and Aerospace Applications

by

Liting Zhou

A thesis submitted to the Graduate Faculty of
Auburn University
in partial fulfillment of the
requirements for the Degree of
Master of Science

Auburn, Alabama
December 15, 2018

Keywords: Deep Learning, Neural Network, Classification, Prediction, Missile Identification and Missile Tracking

Abstract

The ability to rack an enemy missile while in-flight, whether to accurately predict the point of impact (POI), the point of origin (POO) or to destroy the missile in-flight, is greatly enhanced if the missile type and/or physical characteristics (type, size, payload, etc.) are known as a priori. Given the missile type/characteristics, a conditional missile tracking system can be developed using simulated missile fly-outs (6-DOF) based on the physics of that missile type. If the missile type or characteristics are unknown, assuming the known classes of missiles in the enemy's arsenal, a rapid missile classification must be incorporated into the refined posterior tracking system. This tracking system is a two-stage tracking system. In the first stage, within a few milliseconds of radar detection, the missile is rapidly and accurately classified within into one of k classes. In the second stage, the specific tracking system tailored to that specific class is engaged for more accurate tracking.

In this thesis, we focus on deep learning neural networks (DNN) to solve the rapid missile classification problem in this application. We demonstrate the superior performance of DNNs over single layer neural networks, as well as, classical generalized linear model, using 6-DOF-fly-outs of three similar short-range rocket classes. We show that we can achieve 100% corrected classification within milliseconds of flight on our testing data (independent fly-outs).

Acknowledgement

I would like to thank the many people who helped make this master's thesis work in statistics possible. First and foremost, I want to take this chance to thank my excellent advisor, Dr. Mark Carpenter. Your expertise as a teacher, researcher and advisor have helped make my two-year journey into statistics graduate studies productive and fun. I will be forever grateful for the guidance, encouragement and support you gave to me. Watching you work tirelessly, always being my strongest backup, and giving insightful suggestions not only helped me grow as a researcher, but also planted great passion for the art of uncovering the hidden messages in data. To my other committee members, Dr.'s Philippe Gaillard and Peng Zeng, thank you so much for your advice and comments on this work. It has been my honor to have you on my committee and to have had you as teachers, inside and out of the classroom. In addition, I want to thank my R programming mentor, Dr. Betram Zinner, it's you who inspired my love in R programming.

To all my Auburn friends, thank you for making my graduate life so delightful and meaningful here, especially, to my dearest friends, Serge Guerngar, Vicki Miao, Heike Williams, Okey Chidume and Markus Kreitzer. Thank you so much for always being here for me, for without you, I would not be the Lucy I am today. I am so blessed to have you all in my life loving me, caring for me, supporting me and helping me. Thank you to all my colleagues in HudsonAlpha, with special thanks to Dr. Liz Worthey and Matt Holt! Though it was a short two-month internship, I learned so much from you guys, your great passion for science and persistence in discovering the truth in science.

To my parents, Hong Zhou and Xiangsheng Zhou, and my grandpa, Xuedi Zhou, thank you so much! Words cannot express my gratitude for you. Thank you, thank you, thank you! All of you makes Lucy today.

Table of Contents

List of Figures

List of Tables

Chapter 1
Introduction

The main focus in this thesis is the use of deep learning neural networks (DNN) and/or neural networks in aerospace applications. Neural networks (NN) have been used successfully to solve non-linear regression problems (continuous response), as well as, the supervised and unsupervised classification problems (nominal or ordinal responses), across a plethora of fields and applications. For example, Carpenter, Hartfield and Burkhalter (2012) used a NN regression approach in surrogate modeling of various aerodynamic coefficients to speed up the process for reverse-engineering of missiles, in real time. Carpenter, Hartfield and Ahuja (2017) successfully applied ANNS for surrogate modeling of surface vorticity and Carpenter, Hartfield and Zhou (2018) addressed both the rocket identification problem (classification) and the tracking problem (regression).

ANNs can be applied to either multivariate regression problems or the classification problems. In the regression context, neural networks can be viewed as a flexible class of nonlinear regression models, which, as Ripley shows, can be used to approximate any continuous functions, both univariate and multivariate. More specifically, any continuous function $f\colon \mathcal{R}^p \rightarrow \mathcal{R}^k$ can be approximated uniformly on compacta by a general class of functions that includes neural networks and projection pursuit regression (with linear outputs in the hidden layer). Because of this, neural networks are considered to be universal function approximators.

In this thesis, we focus on deep learning neural networks (DNN) to solve the rapid missile classification problem in aerospace applications. In particular, we show that DNNs improve over the neural network approach used Carpenter, Hartfield and Speakman (2016). We demonstrate the superior performance of DNNs over single layer neural networks, as well as, classical generalized linear model, using 6-DOF-fly-outs of three similar short-range rocket classes. We show that we can achieve 100% correct classification within milliseconds of flight on our testing data (independent fly-outs).

The ability to track an enemy missile while it is in-flight, whether to accurately predict the point of impact (POI), point of origin (POO) or to destroy the missile in-flight, is greatly enhanced if the missile type and/or physical characteristics (type, size, payload, etc.) are known a priori. Given the missile type/characteristics, a conditional missile tracking system can be developed using simulated missile fly-outs (6-DOF) based on the physics of that missile type. If the missile type or characteristics are unknown, assuming the known classes of missiles in the enemy's arsenal, a rapid missile classification must be incorporated into the refined posterior tracking system. This tracking system is a two-stage tracking system. In the first stage, within a few milliseconds of radar detection, the missile is rapidly and accurately classified within into one of k classes. In the second stage, the specific tracking system tailored to that specific class is engaged for more accurate tracking.

In Chapter 2, we describe the concepts behind Deep Learning, In Chapter 3, we introduce the aerospace application that we focus on in this thesis and the associated data. In Chapter 4, we show the results. Lastly, in Chapter 5, we describe further research.

Chapter 2

Deep Learning

The main statistical learning topic in this thesis is Deep Learning (DL) or Deep Neural Networks (DNN). Deep Learning is a machine learning algorithm designed to mimic the structure and function of the brain in the form of artificial neural networks. In other words, DNNs are feedforward, artificial neural networks (ANN) with multiple layers of non-linear hidden units, see Figure 1 for an illustration. Over the past few years, developments in machine learning algorithms and computer hardware have led to various deep learning methods of learning feature hierarchies such as a wide array of deep architectures including neural networks with many hidden layers (Vincent et al., 2008) and graphical models with many levels of hidden variables (Hinton et al, 2006). According to the theoretical results reviewed by Bengio (2009),in order to learn the kind of complicated functions that can represent high-level abstractions, deep architectures may be needed. And Rumelhart (1986) showed the experimental results with deep architecture can be obtained with deep neural networks model.

In this modern “big-data” era, data sets are not only growing in terms of the number of observations, but in terms of the number of the dimensions of the feature space, so finding the optimal output representation with a shallow model of Machine Learning algorithms using one or two layers of data transformation is not always possible. In these situations, DL provides a multi-layer approach to learn data representations, typically performing with a multi-layer neural network. Deep learning approaches such as Deep Feedforward Neural Network (DFNN), Recurrent Neural Network (RNN) and Convolutional Neural Network (CNN) have been proved highly successful in predicting a wide variety of problems (i.e. image classification, pattern recognition, sequential tasks, object detection, speech recognition, autonomous driving, etc.). And feedforward neural networks are especially widely used in classification problems.

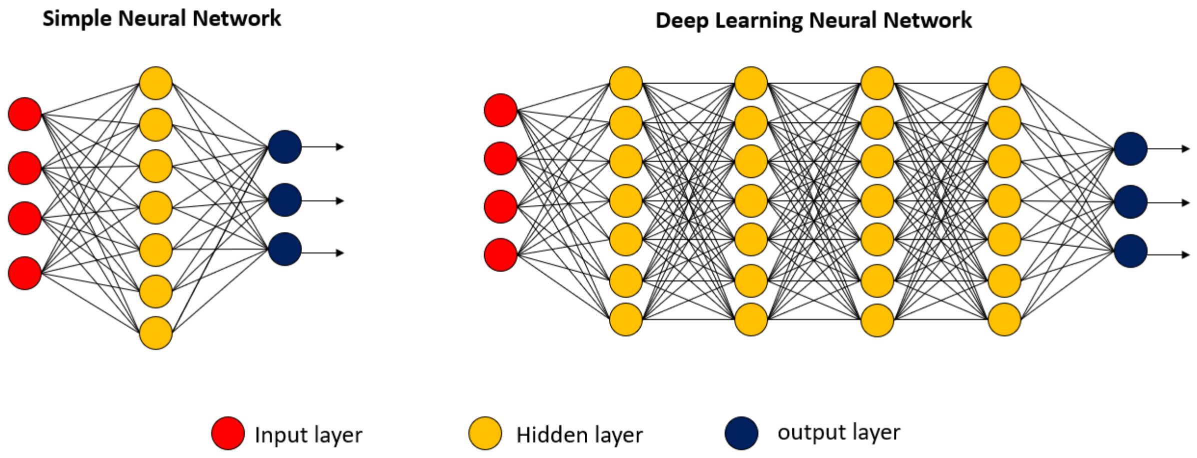


Figure 1: Illustration of the basic difference between NN and DNNs.

In this chapter, we describe Neural Networks (NN) with Deep Neural Networks (DNNs). In Sections 2.1-2.3, of this chapter, we describe the basic NN in detail, including the mathematics, activation functions, and the algorithms for training these simple neural network, respectively. In Section 2.4, we describe the deep learning neural network (DNN).

2.1 Basic Neural Network

In this section, we describe the basic neural network and all of its various components. In Section, 2.1.1., we describe the basic structure or architecture of a feed-forward, single hidden layer, neural network with back-propagation and, in Section 2.1.2., we go into detail about the different activation functions.

2.1.1 Neural Network structure

A Neural Network, in general, is made of the following parts: input layer, hidden layer and output layer. For basic neural network, it only has one input layer, one hidden layer and one output layer. All units in the neural network are also called neurons. All input variables (input neurons) are fed to the input layer and the output layer is composed of all output variables (output neurons). The units in hidden layers are hidden neurons, they are derived from applying non-linear function (activation) of a linear combination of its inputs.

The Figure 2 shows a neural network for classification with one input layer (8 input neurons), one hidden layer (4 hidden neurons) and one output layer (4 output neurons).

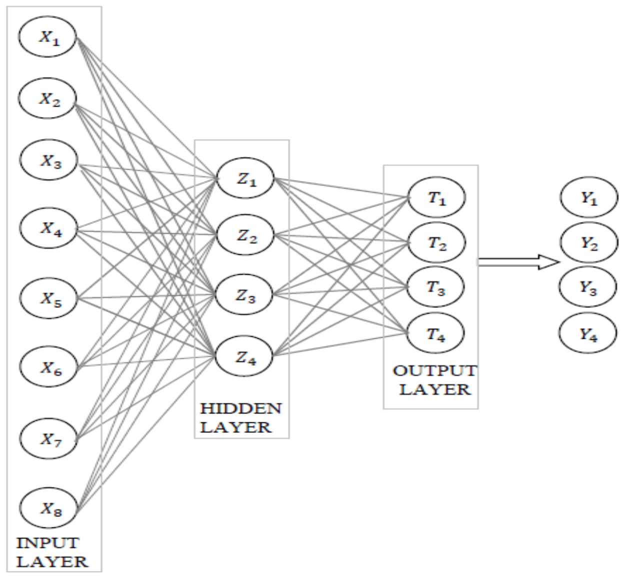


Figure 2: Neural Network with one hidden layer (8:4:4)

The neural network in Figure 2 is a classic feedforward neural network. And it can be mathematically generalized by following notations in Hastie, et al. (2008). In the Figure 2, we are showing a 4-class classification since there are 4 output neurons. There are 4 target measurements Y_k, $k = 1,2,3,4$, each being coded as a 0-1 variable for the 4th class. There are 8 input neurons X_n $(n = 1, \dots ,8)$ in the input layers.

Derived features Z_m $(\mathrm{m} = 1, \dots , M)$ are linear combinations of the 8 inputs which are "activated" by the activation function $\sigma(.)$. The targets Y_k $(k = 1,2,3,4)$ are the linear combinations of the derived features Z_m $(\mathrm{m} = 1, \dots , M)$.

$$\begin{aligned} &Z_m = \sigma(\alpha_{0m} + {\alpha_m}^T X), m = 1, \dots , M, \\ &T_k = \beta_{0k} + {\beta_k}^T Z, k = 1, \dots ,4, \qquad (2.1.1) \\ &f_k(X) = g_k(T), k = 1, \dots ,4, \end{aligned}$$

where $Z = (Z_1, Z_2, \dots , Z_M)$ is the derived features vector, $X = (X_1, X, \dots , X_8)$ is the input vector, T = $(T_1, T_2, \dots , T_4)$ is the output vector referred to as the target vector. The function $g_k(\cdot)$ is the identity function of neural nets taking nonlinear regression approach, it gives a final transformation of the output vectors T.

2.2 Neural Network Activation

An activation function is actually a non-linear transformation function that computes a "weighted sum" of its inputs, adds a bias and then decides whether the information that the neuron is receiving is relevant for the given information or should it be ignored.

$$Y = \Sigma(weight \ \times input) + bias$$

Here, the range of the value Y is $(-\infty, +\infty)$. The standard of taking the information into account or not depends on different activation function, such as sigmoid, tanh and ReLU. Figure 3 illustrates a sigmoid function $= \frac{1}{1+e^{-x}}$.

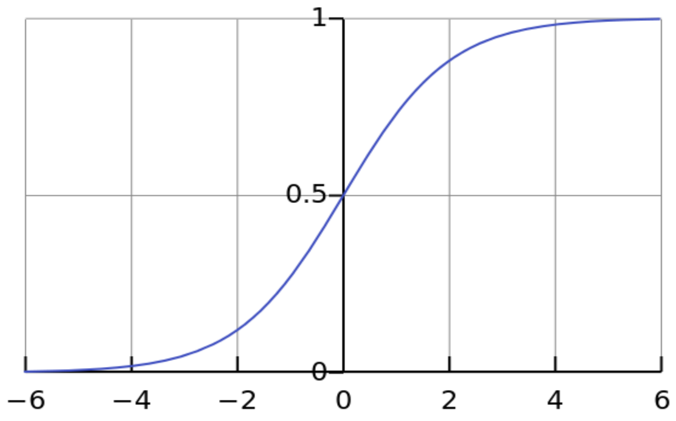


Figure 3: Sigmoid activation function

As illustrated, we see that the sigmoid function is nonlinear, and any combinations of this function will still be nonlinear. From the above Figure 3, it’s easy to find out any small changes in X will lead a big change in Y which would give a good shot for classification prediction. However, Y would respond less to changes in X when Y is going toward to the end. This indicates that the gradient in that area will be small, which will bring the problem of “disappearing gradients” letting network refuse to learn further.

For Tanh function $f(x) = \tanh(x) = \frac{2}{1+e^{-2x}} - 1$:

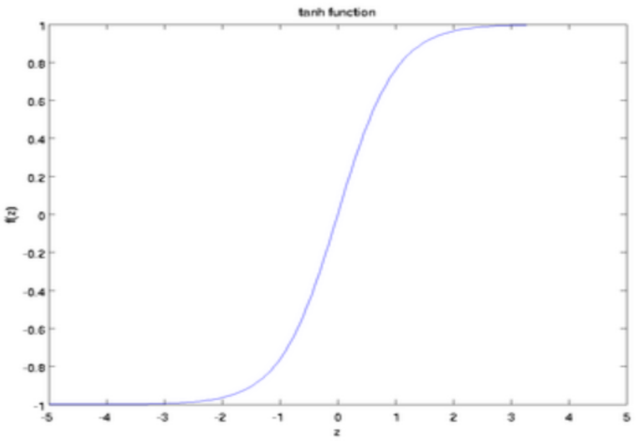


Figure 4: Tanh activation function

The tanh function is actually just a scaled sigmoid function, since $\tanh(x) = 2\ sigmoid(2x) - 1$, and it is also nonlinear. Compared with sigmoid function, even though the gradient is stronger because of its steeper derivatives, it still has the vanishing gradient problem.

For ReLU function (Rectified Linear Unit), $f(x) = \max(\ 0, x\ )$:

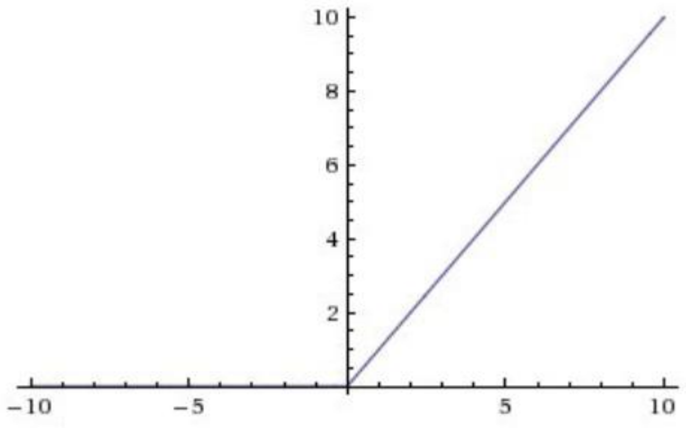


Figure 5: ReLU function

The range of ReLU function is $[0, \infty)$, which means it only gives an output if x is positive, otherwise 0. Although it looks like linear, it is nonlinear in nature. The benefit of running ReLU activation function is due to its sparsity. If we are given a big neural network with a lot of neurons, using sigmoid or tanh will be very costly since they will cause activations to be processed to describe the output of a network by using almost all neurons. It is because ReLU only allows positive values to pass that it will definitely speed up the process and make the activations sparse and efficient. Therefore, ReLu tends to provide a lighter network with fewer neurons so that it can bring down the possibility of occurrence of a dead neuron.

In summary, Sigmoid function will work better if classification is binary, tanh functions are not widely used because of the dead neuron problem, and ReLU is one of the most widely used activation function and usually yields better results compared with Sigmoid and Tanh.

2.3 Neural Network Backpropagation

The unknown connecting parameters between layers are often called weights, and these values are produced for modelling fit. For the neural net given above, it consists of M(n+1) + K(M+1) weights. We assume θ be the vector of length M(n+1) + K(M+1) containing the weights

$$\{\alpha_{0m}, \alpha_m, m = 1,2,\ldots,M\} \text{ and } \{\beta_{0k}, \beta_k, k = 1,2,\ldots K\}, where\ K = 4. \qquad (2.1.2)$$

For regression problem, we use sum-of-squared errors to be the error function measuring fit:

$$R(\theta) = \sum_{i=1}^{N}\sum_{k=1}^{K}(y_{ik} - f_k(x_i))^2, \qquad (2.1.3)$$

where N is the total number of observations.

For classification problem, we use squared error or cross-entropy (deviance):

$$R(\theta) = -\sum_{i=1}^{N}\sum_{k=1}^{K} y_{ik} \log f_k(x_i)\ , \qquad (2.1.4)$$

where N is the total number of observations.

In this way, the classifier will be $G(x) = argmax_k f_k(x)$.

In neural network, gradient descent called *back-propagation* is commonly used to minimize $R(\theta)$. Actually, the process of how neural networks assess its own accuracy and automatically adjust the weights across all the node connections to try to improve that accuracy is called backpropagation. Generally, the backpropagation is used to train Deep Neural Network.

Assume $z_{mi} = \sigma(\alpha_{0m} + {\alpha_m}^T x_i)$ and $z_i = (z_{1i}, z_{2i}, \dots, z_{Mi})$, the backpropagation for squared error loss will be:

$$R(\theta) \equiv \sum_{i=1}^{N} R_i = \sum_{i=1}^{N}\sum_{k=1}^{K}(y_{ik} - f_k(x_i))^2, \tag{2.1.5}$$

with derivatives

$$\begin{aligned} \frac{\partial R_i}{\partial \beta_{km}} &= -2\big(y_{ik} - f_k(x_i)\big)g'_k\big({\beta_k}^T z_i\big)z_{mi}, \\ \frac{\partial R_i}{\partial \alpha_{ml}} &= -\sum_{k=1}^{K} 2\big(y_{ik} - f_k(x_i)\big)g'_k\big({\beta_k}^T z_i\big)\beta_{km}\sigma'({\alpha_m}^T x_i)x_{il}\,. \end{aligned} \tag{2.1.6}$$

Based on these derivatives, the new gradient descent at the (r+1)st iteration will become:

$$\begin{aligned} {\beta_{km}}^{(r+1)} &= {\beta_{km}}^{(r)} - \gamma_r \sum_{i=1}^{N} \frac{\partial R_i}{\partial {\beta_{km}}^{(r)}}, \\ {\alpha_{ml}}^{(r+1)} &= {\alpha_{ml}}^{(r)} - \gamma_r \sum_{i=1}^{N} \frac{\partial R_i}{\partial {\alpha_{ml}}^{(r)}}, \end{aligned} \tag{2.1.7}$$

where γ_r is the learning rate, which is:

$$\begin{aligned} \frac{\partial R_i}{\partial \beta_{km}} &= \delta_{ki} z_{mi}, \\ \frac{\partial R_i}{\partial \alpha_{ml}} &= s_{mi} x_{il}\,. \end{aligned} \tag{2.1.8}$$

The δ_{ki} and s_{mi} are the “errors” in the hidden layer and output layer units from the present model respectively. By definitions, these errors satisfy

$$s_{mi} = \sigma'({\alpha_m}^T x_i) \sum_{k=1}^{K} \beta_{km} \delta_{ki}, \tag{2.1.9}$$

which is known as backpropagation equations. The new gradient descent can be calculated with backpropagation (also known as two-pass algorithm procedure) applying this equation. In the forward pass, using the formula (2.1.1), the fixed weights and predicted values $\widehat{f_k}(x_i)$ can be given. In the backward pass, we compute errors δ_{ki} first, and then get the errors s_{mi} through (2.1.9). Then, we apply both errors into the equation (2.1.8) to compute the gradients for the updates in (2.1.7).

2.4 Deep Neural Network

As mentioned previously, a deep neural network is an ANN with multiple hidden layers. By increasing more hidden units and layers, it can help to run a deep and highly parameterized neural networks to improve the prediction performance. Especially for the high dimensional data.

Not like machine learning, features of data have to be defined before modeling, in Deep Neural Networks, the hidden layers offer the auto-identify features to the means. The structure of deep neural network is shown as below:

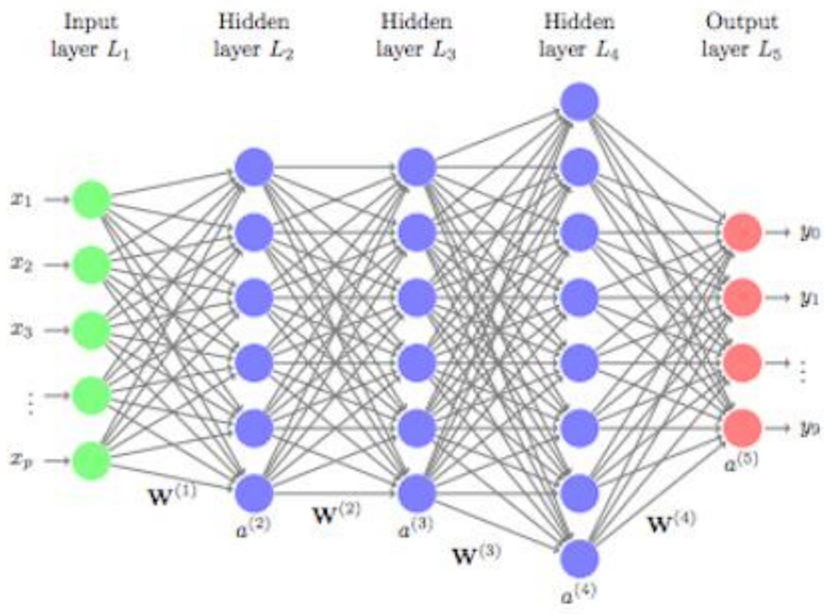


Figure 6: Deep feedforward neural network

Similar to the regular neural networks, deep feedforward neural network is also mainly composed of three parts: one input layer($x_1, x_2, \dots, x_p$), multiple hidden layers, and one output layer($y_1, y_2, \dots, y_9$). $W(\cdot)$ is the weight function adding weight from each neuron in the last layer to each neuron in the next layer. Because Deep Neural Networks can perform successive non-linear transformations across each layer which equips itself with a strong ability to model very complex and non-linear relationship. DNNs is also fit for solving traditional regression and classification problems. Especially, when the dimensions of the data are very large. However, if the number of observations and feature inputs decreases, traditional shallow machine learning methods and regular neural network tend to perform just as well, sometimes, even more efficient.

In deep neural networks, there are many DNN extensions models. For instance, Deep Feedforward Neural Networks (DFNN) is extensively used for pattern recognition and visual classification tasks and object detection. Convolutional neural networks (CNN) is often used for image/video recognition, recurrent neural networks (RNN) is commonly used for pattern recognition and sequential tasks like time series prediction, sequence labeling and sequence classification. Long short-term memory neural networks (LSTM) are advancing automated robotics and machine translation.

However, overfitting is a serious problem in such networks since it is hard for large networks to deal with overfitting by combining the predictions of many different large neural nets at test time. Geoffrey Hinton, et.al., (2014) found out that Dropout is a useful technique for addressing Neural Networks' overfit problem. Generally, Dropout can sample from an exponential number of different "thinned" networks during training. And when we are running the model for test dataset, it can approximate the effect of averaging the predictions of all these thinned networks by simply using a single untinned network that has smaller weights. This process has been proven that it is able to reduce overfitting significantly and give major improvements over other regularization methods.

Chapter 3

Application of DNN in Aerospace

Rapid Projectile Classification in Support of Missile Tracking

As we mentioned in the introductory chapter, in this thesis, we focus on the projectile (munitions, rockets, missiles) classification problem. The ability to track an enemy missile while it is in-flight, whether to accurately predict the point of impact (POI), point of origin (POO) or to destroy the missile in-flight, is greatly enhanced if the missile type and/or physical characteristics (type, size, payload, etc.) are known a priori. Given the missile type/characteristics, a conditional missile tracking system can be developed using simulated missile fly-outs (6-DOF) based on the physics of that missile type. If the missile type or characteristics are unknown, assuming the known classes of missiles in the enemy's arsenal, a rapid missile classification must be incorporated into the refined posterior tracking system. This tracking system has two stages. In the first stage, within a few milliseconds of radar detection, the missile is rapidly and accurately classified within into one of k classes. In the second stage, the specific tracking system tailored to that specific class is engaged for more accurate tracking.

For our application we decided to work with the classification of short range munitions. Figure 7 displays images of three rockets/RAM-threats for which we demonstrate our approach in this paper, along with their physical descriptions. These munitions are the types of munitions that are often shot from on enemy camp to the other. During the height of the war in Iraq, our US bases places were constantly under attack with these types of weapons. The US forces would literally like to have the capability to detect, identify and either shoot these out of sky before they hit on the base or at least predict the point of impact (POI). Also, they may want to “reverse-track” these munitions to determine the point of origin (POO), so that they may target the enemy.

107 mm Rocket	122 mm Rocket	70 mm Rocket
Length = 0.76 m.	Length = 2.87 m.	Length = 1.06 m.
Launch mass = 18.75 kg.	Launch mass = 65.6 kg.	Launch mass = 10.43 kg.
Burnout mass = 15.3 kg.	Burnout mass = 45.2 kg.	Burnout mass = 7.14 kg.
Average thrust = 11,615 N.	Average thrust = 21,525 N.	Average thrust = 5,841 N.
Action time = 0.6 sec.	Action time = 1.8 sec.	Action time = 1.1 sec.
Burnout speed = ~ 400 m/s	Burnout speed = ~ 700 m/s	Burnout speed = ~ 700 m/s
Max. range = ~ 8.5 km.	Max. range = ~ 21 km.	Max. range = ~ 8.0 km.
TOF = ~ 46 sec.	TOF = ~ 75 sec.	TOF = ~ 52 sec.

Figure 7: Description of Example Threat/Target Physical Parameters used in this paper. It is clear that the 70 mm and 107 mm rockets have similar physical characteristics, as well as, expected flight performances, but the 122 mm rocket is much bigger than both with typically longer range and time of flight.

It is clear that the 70 mm and 107 mm rockets have similar physical characteristics, as well as, expected flight performances, but the 122 mm rocket is much bigger than both with typically longer range and time of flight. Figure 8 provides illustrations typical flight trajectories for the 70, 107 and 122 mm rockets using in this paper. The upper left panel shows the range over time for three simulated rockets, the upper right shows the speed over time, the lower left shows the altitude over time and the lower right the altitude over the downrange values.

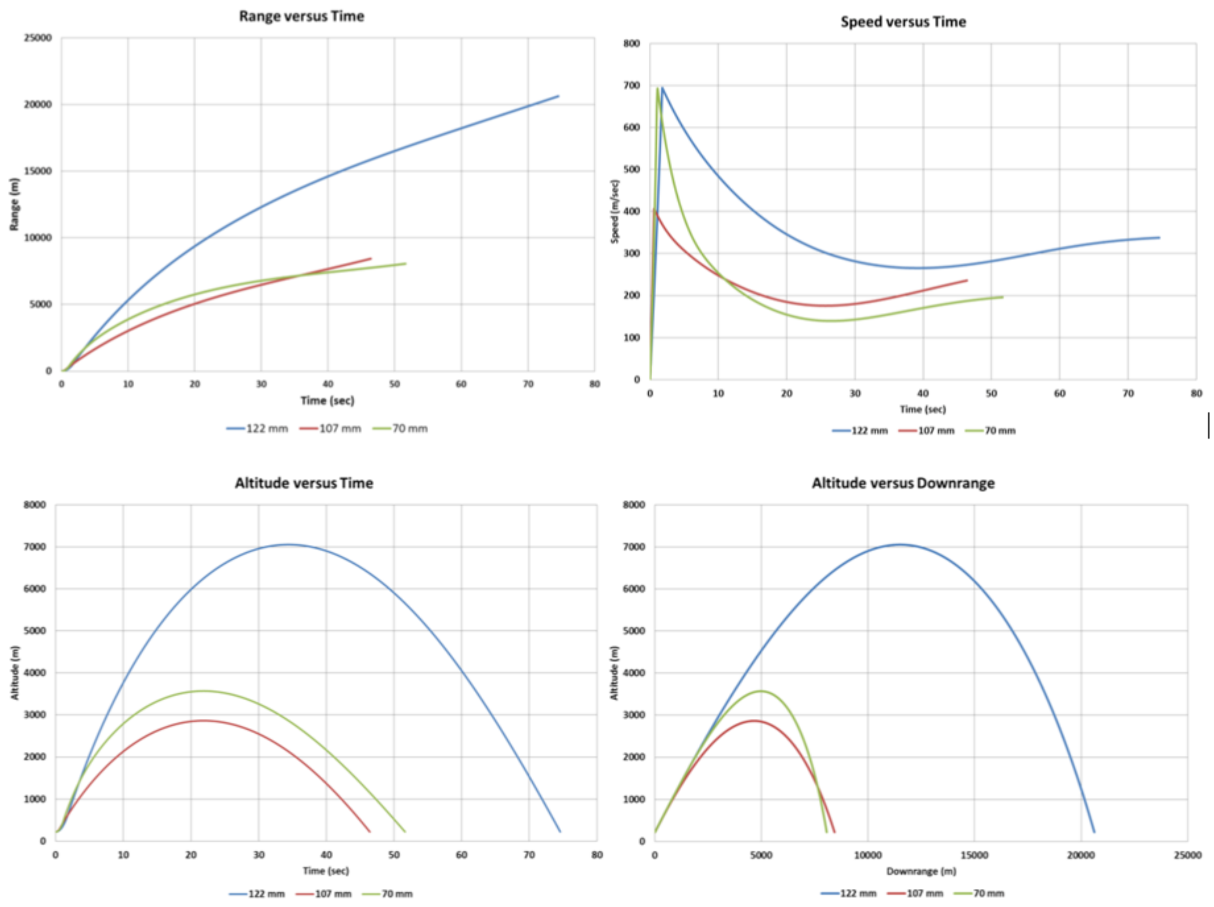


Figure 8: Example trajectory data for the 70, 107 and 122 mm rockets using in this paper. The upper left panel shows the range over time for three simulated rockets, the upper right shows the speed over time, the lower left shows the altitude over time and the lower right the altitude over the downrange values.

When we set out to work on this problem, the goal was to do threat-type classification and identification employing conventional radar measurement of the threat's position in the radar frame transformed to Cartesian coordinates, which represent cross range, down range and altitude. The radar data are converted to Cartesian ENU coordinates. These coordinates are rotated and translated to match coordinate system in the training data (3DOF); see Figure 9. The converted trajectories will serve as the input into the previously trained and validated Neural Net Prediction models while Target type and caliber are predicted. No additional data input sources or signal processing ~~are~~ is required. The neural network approach can provide, valuable threat information such as point-of-origin (POO) estimation, point-of-impact (POI) prediction and prediction of type/caliber.

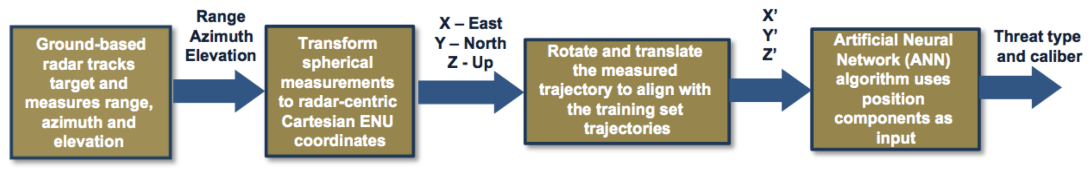
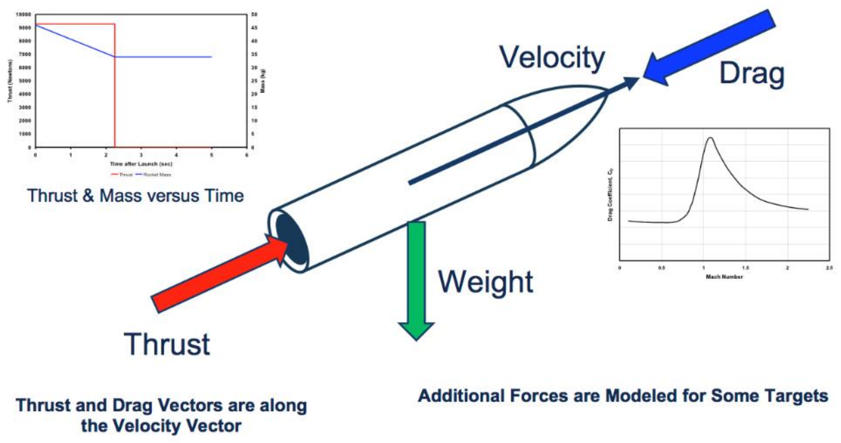

Ground-based radar tracks target and measures range, azimuth and elevation
Range Azimuth Elevation
Transform spherical measurements to radar-centric Cartesian ENU coordinates
X – East Y – North Z - Up
Rotate and translate the measured trajectory to align with the training set trajectories
X' Y' Z'
Artificial Neural Network (ANN) algorithm uses position components as input
Threat type and caliber

Figure 9: Graphical representation of how the raw radar data is converted to inputs. Radar data converted to Cartesian ENU coordinates. The coordinates are rotated and translated to match coordinate system in the training data (3DOF). The converted trajectories, (x,y,z,time), are input into the previously trained and validated NN prediction models

Simulated Data for Fly-outs (3DOF): Obviously, for this project, we cannot actually obtain real radar data on real rocket fly-outs so the data we used was simulated (3DOF) data that we obtain from Dr. Hartfield in the Department of Aerospace Engineering. The three degree-of-freedom (3DOF) simulations are based on equations of motion from fundamental physics, translational degrees-of-freedom only. The motion of the mass center is simulated. Assumes a flat, non-rotating, constant gravity earth model. The position, velocity and acceleration vectors are calculated. The parameters required to model threat target are, thrust vs. time (rockets), V0 (mortars), mass vs. time, drag coefficient vs. Mach number. The Simulation is stochastic via randomization of parameter can be randomized. These simulations are based on validated using actual measured trajectories. See Figure 10 for graphical description of the forces acting on the threat target, modeled in the 3DOF simulations, and Figure 11 for a graphical description of this process.



Thrust & Mass versus Time
Velocity
Drag
Weight
Thrust
Thrust and Drag Vectors are along the Velocity Vector
Additional Forces are Modeled for Some Targets

Figure 10: Forces acting on the threat target which are modeled in our 3DOF simulations. The motion of the mass center is simulated. Assumes a flat, non-rotating, constant gravity earth model. The position, velocity and acceleration vectors are calculated. The parameters required to model threat target are, thrust vs. time (rockets), V0 (mortars), mass vs. time, drag coefficient vs. Mach number. The Simulation is stochastic via randomization of parameter can be randomized. These simulations are based on validated using actual measured trajectories

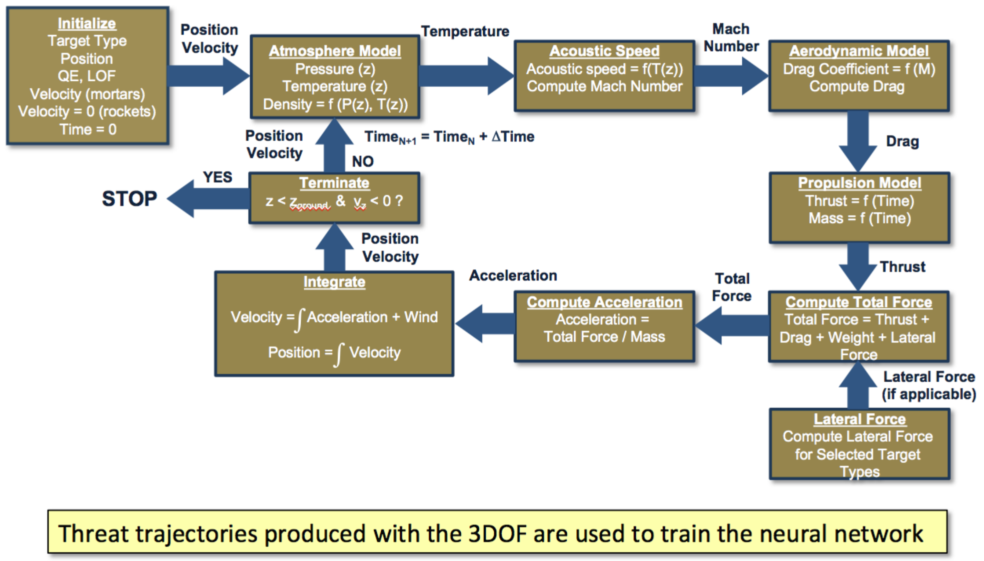


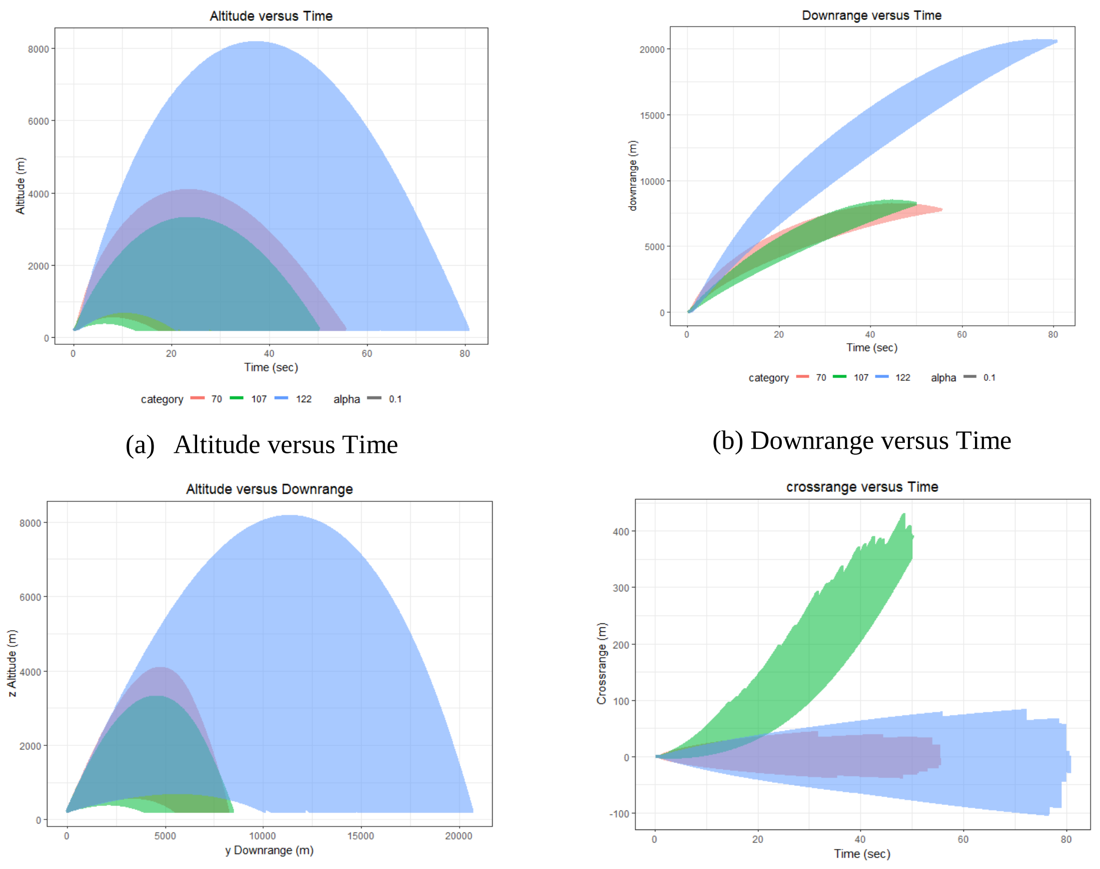
Figure 11: Threat target dynamic three degree-of-freedom (3DOF) simulation. The three degree-of-freedom (3DOF) simulation are based on equations of motion from fundamental physics, translational degrees-of-freedom only.

In Section 3.1., we provide the background information and data description for this application. In Section 3.2., we describe the training, validation and testing data sets.

3.1 Data Description

The data were imported into R studio Version 1.0.153 and analyzed using various procedures in R. There are 1194864 observations (for three missiles in total) in our stimulated missile radar data. The data has 1500 simulated trajectories/flights, 500 trajectories for each caliber (70mm, 107mm and 122 mm), and every observation has the conventional radar measurements of the threat's position in the radar frame transformed to Cartesian coordinates (cross rage, down rage and altitude). The converted trajectories, (x, y, z, time) are the inputs in our model,

Our goal is to classify three different types of rocket in less than 1 to 2 seconds. The graph below shows the basic information about these three rockets:



(a) Altitude versus Time

(b) Downrange versus Time

(c) Cross range versus Time

(d) Altitude versus Downrange

Figure 12: Basic trajectory information about three rockets

From the figure 12, we can see that 70s and 107s tend to have very similar flight trajectories. There is also a great amount of variability for the 122s.

Before we start the analysis, we divide the data from each class into three data sets for training, validation and test respectively. We use the training data set to build the predictive models, and during the training process, we evaluate the models using validation data set. Once a final model has been selected, this model will be tested using the independent data “test dataset” .The figure below graphically describes this process for an experiment with 3 classes of missile and 500 fly-outs for each class.

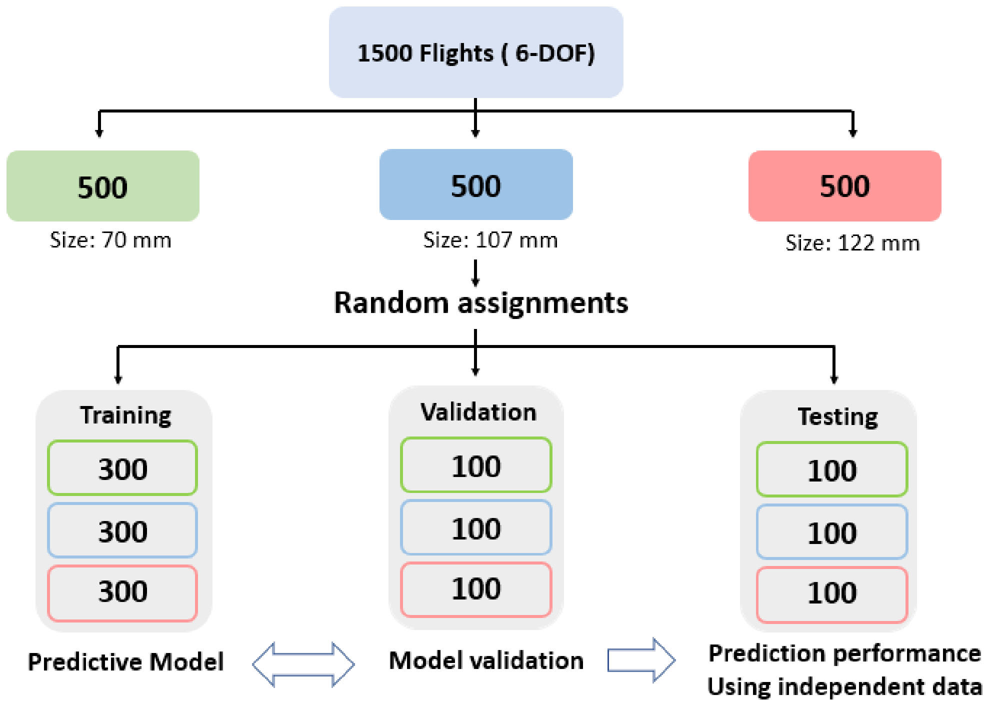
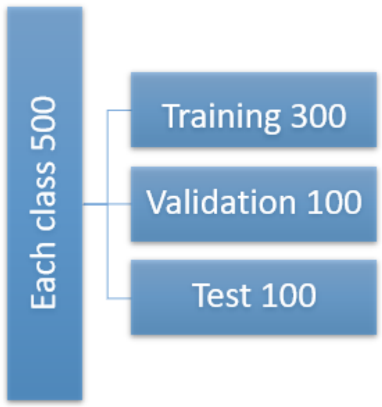
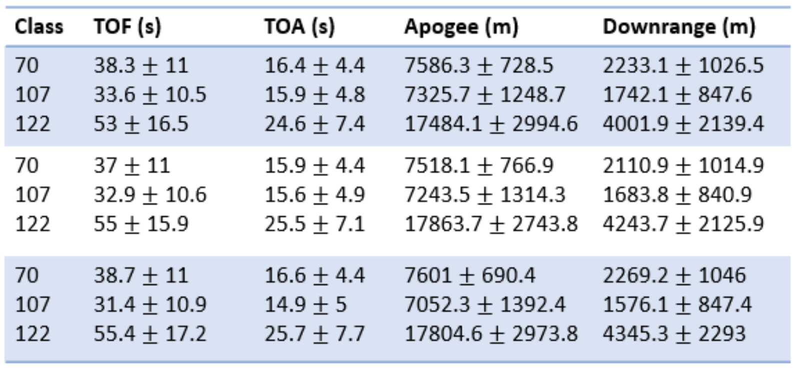


Figure 13: Process of data management.

After we partitioned the data, the summary for the datasets is shown in the below figure.





Class	TOF (s)	TOA (s)	Apogee (m)	Downrange (m)
70	38.3 ± 11	16.4 ± 4.4	7586.3 ± 728.5	2233.1 ± 1026.5
107	33.6 ± 10.5	15.9 ± 4.8	7325.7 ± 1248.7	1742.1 ± 847.6
122	53 ± 16.5	24.6 ± 7.4	17484.1 ± 2994.6	4001.9 ± 2139.4
70	37 ± 11	15.9 ± 4.4	7518.1 ± 766.9	2110.9 ± 1014.9
107	32.9 ± 10.6	15.6 ± 4.9	7243.5 ± 1314.3	1683.8 ± 840.9
122	55 ± 15.9	25.5 ± 7.1	17863.7 ± 2743.8	4243.7 ± 2125.9
70	38.7 ± 11	16.6 ± 4.4	7601 ± 690.4	2269.2 ± 1046
107	31.4 ± 10.9	14.9 ± 5	7052.3 ± 1392.4	1576.1 ± 847.4
122	55.4 ± 17.2	25.7 ± 7.7	17804.6 ± 2973.8	4345.3 ± 2293

Figure 14: Case study summary. In this figure, TOF stands for Time Of Flight, TOA represents Time Of Apogee, and Apogee is the maximum value of altitude for all flights. This figure shows us that 122 mm rockets fly longest and highest , at the meantime, it also flies furthest. Besides, it's also easy to see, not only 70mm and 107 mm rockets have similar physical characteristics (which we know from the figure 7), they also have the similar time of flight and apogee. Also, when the size is getting bigger, it's not necessary that the further and higher the rockets will fly. The above figure clearly indicates that 70mm rockets fly higher and further compared with 107mm rockets.

3.2 Neural Network

3.2.1 Analysis Workflow

Put cross range (x), downrange(y), altitude (z) and time in the input layer, feed three classes: rocket 70mm, 107mm and 122 mm into the output layer.

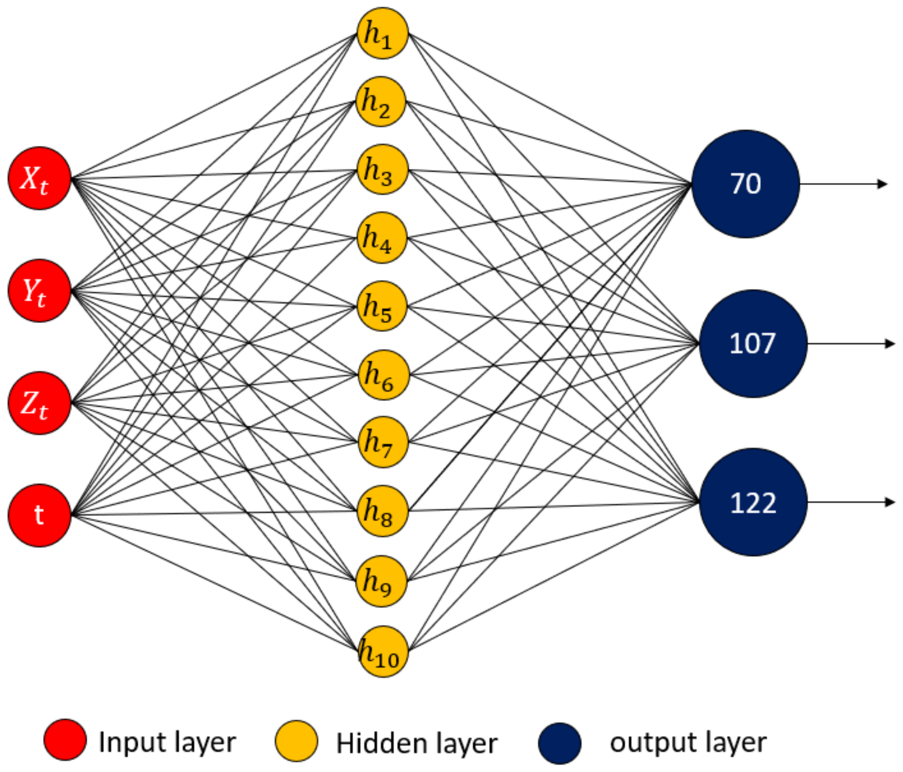


Figure 15: Regular Neural Network architecture

3.2.2 Fitting Neural Network

I use R package "nnet" to construct standard neural network with one hidden layer of sigmoid function neurons. There are several parameters which have always provided better results when changed from their defaults. One is to add a small decay to the weights, so they decrease over time unless reinforced by new data. The second is to increase the maximum number of iterations before training halts, from 100 to 100000.

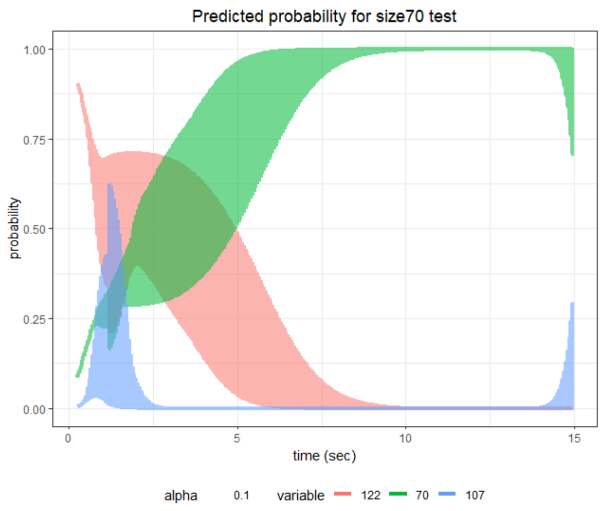
In addition, there is another very vital parameter, the number of hidden neurons to take in the hidden layer. There are some thumb rules that people use, such as never more than twice the number of inputs or 1/30 of your training samples, but these are not specific enough to reliably give a good result. Based on our data, I tried several numbers.

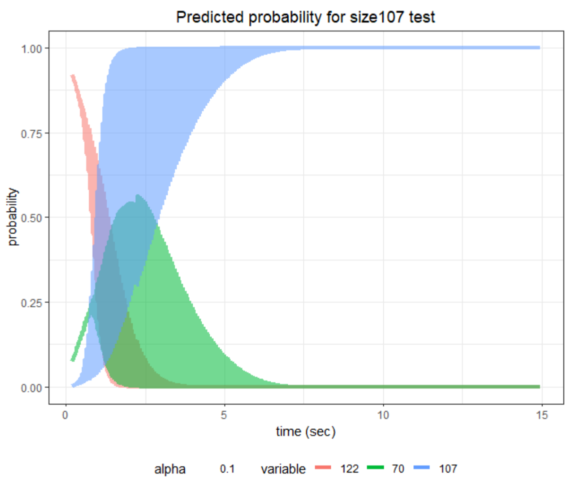
The table below shows the prediction accuracy on the test data with time less than 15 seconds.

	Size 70	Size107	Size 122
Hidden units 10	98.75%	99.41%	98.96%
Hidden units 15	95.03%	98.47%	96.47%
Hidden units 120	99.55%	98.97%	98.83%

Table 1: Prediction accuracy based on test data within 15 seconds

Below are the fitted predicted probability plots for each of the three missile calibers, based on a feed-forward neural network with one hidden layer under 15 seconds. The plots represent the computed probabilities as a function of time on the independent Testing data.





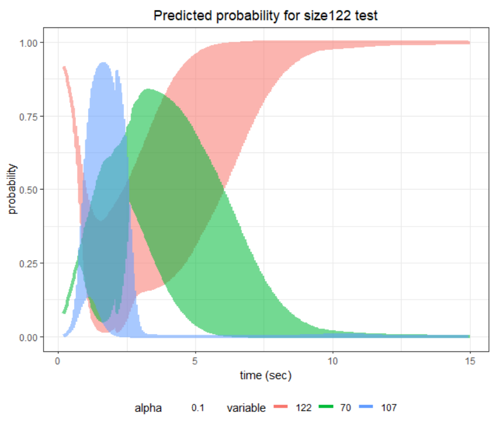


Figure 16: Regular Neural Network predicted probability plots under 15 seconds

According to the above plots, we can see it's hard to 100% classify the missile in the first 5 seconds based on one-hidden layer neural network.

Here are the probability plots after running one-hidden layer neural network with 30 seconds.

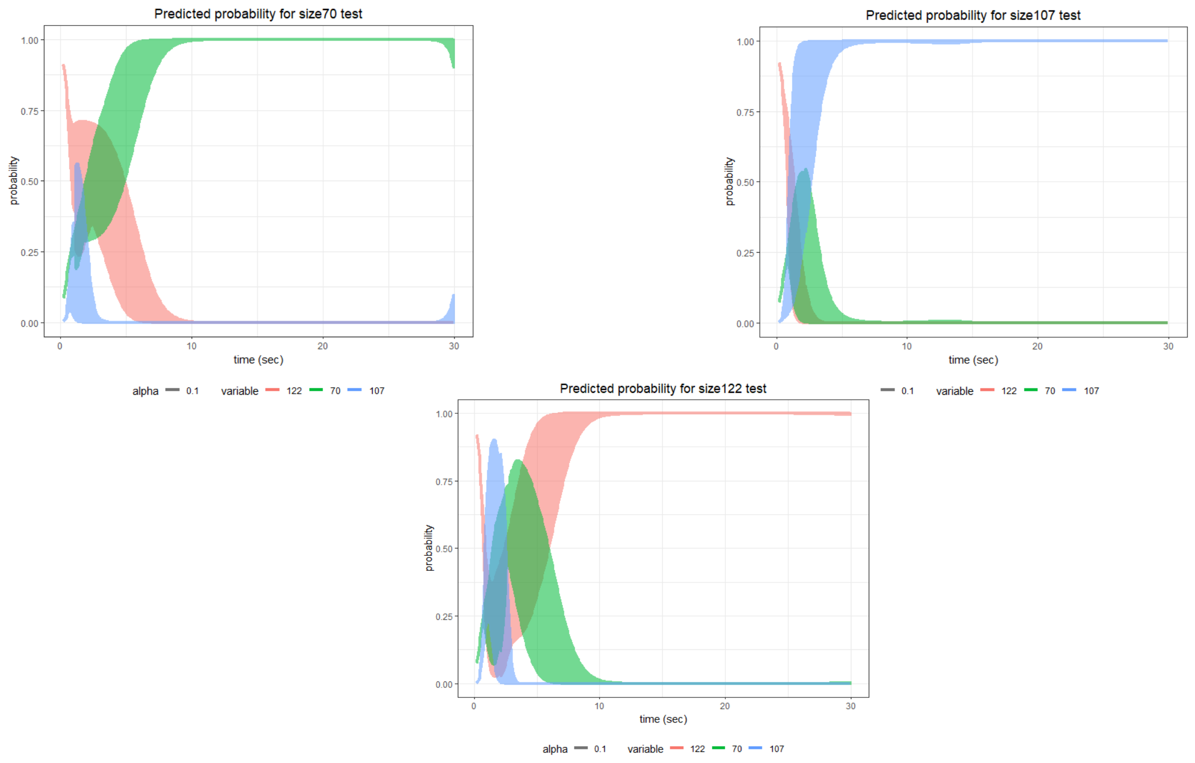


Figure 17: Neural Network predicted probability plots under 30 seconds

The above plots indicated that based on the one-hidden layer neural network with 30 seconds, we can start classifying three calibers 100% correctly after 6-7 seconds.

3.3 Deep Neural Network

3.3.1 Deep Neural Network workflow

Among all neural network approaches in deep learning, we choose deep feed-forward neural network to improve the prediction accuracy for our missile radar data. We build deep neural network in R using Package " keras".

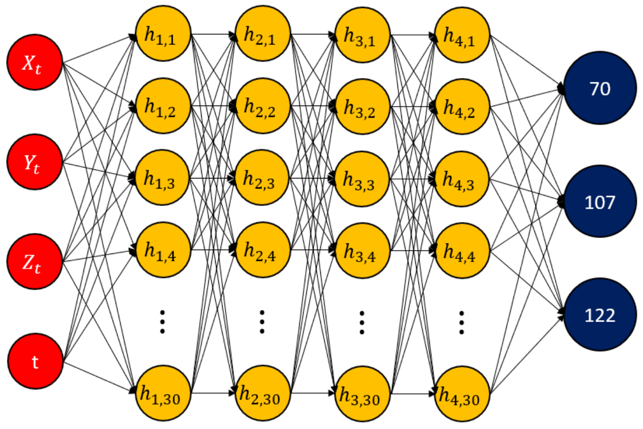


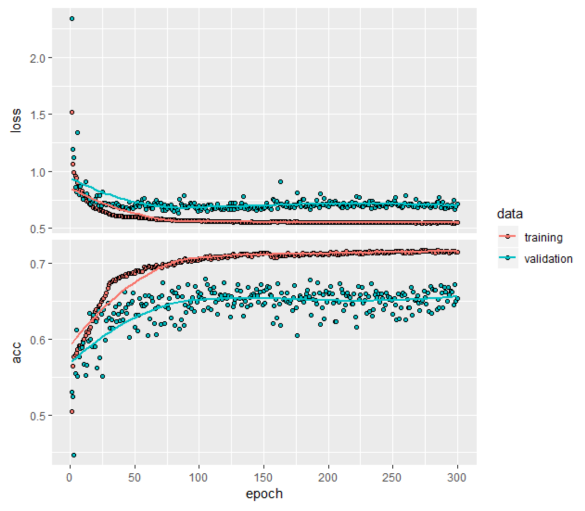
Figure 18: Deep Neural Network architecture

Before we decide on the four hidden layers, we compared the neural networks with different hidden layers, according to the thumb rules, deep neural network usually choose 4-30 hidden layers. The process of choosing number of hidden layers will be shown in detail in next session.

3.3.2 Fitting deep feed-forward neural network

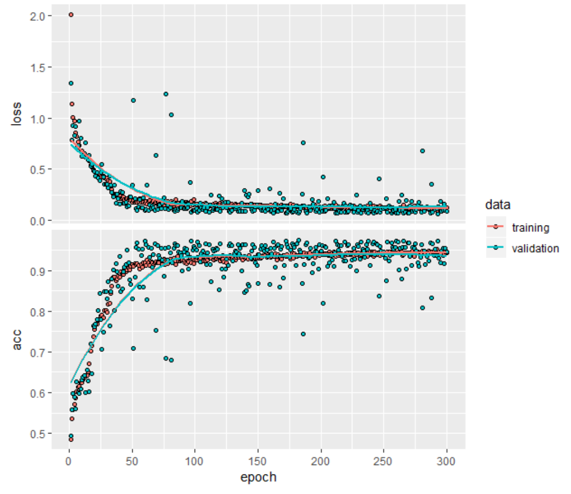
To find an optimal DNN model, the first task is to tune different parameters, such as increasing epochs. Epoch describes the number of times the algorithm sees the entire dataset. In another word, the more complex the features and relationships in the data, the more epochs you will require for your model to learn, adjust the weights and minimize the loss function.

The below plots show how the loss function improve for each epoch based on the deep feed-forward neural network with ReLU activation function.



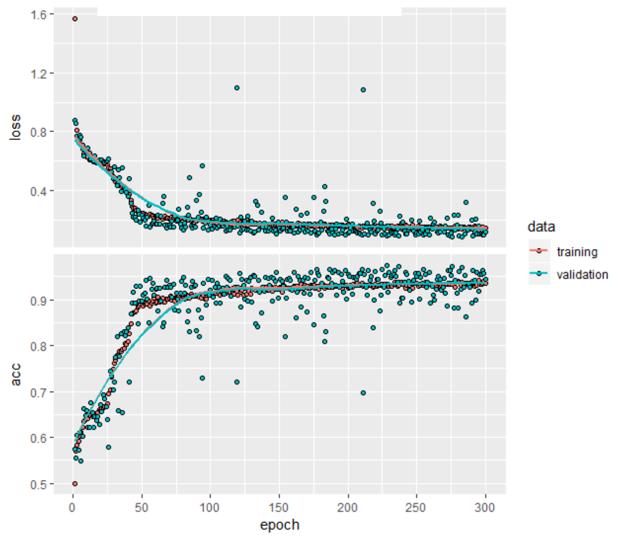
H = 10,10

Accuracy: 91.18%



H = 60,60

Accuracy: 95.07%



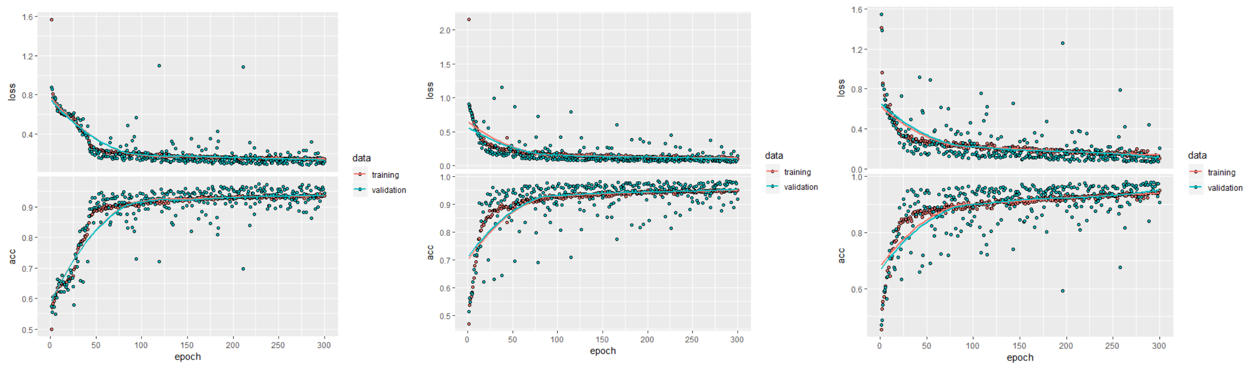
H = 30,30

Accuracy: 94.55 %

Figure 19 Deep Neural Network probability plots varying from number of hidden units

According to the above plots, we see that the accuracy will get better when the number of hidden units is increasing. Even though it looks hidden units 60 gives a better accuracy than 30, but since it will cost hidden units 60 more time to finish modeling, and the accuracies are similar to each other, we choose 30 as our number of hidden neurons for deep feed-forward neural network to classify the three calibers.

After we choose hidden units 30, then we need to decide on the number of hidden layers.



Hidden 30,30
Accuracy: 94.55 %

Hidden 30,30,30
Accuracy: 95.13 %

Hidden 30,30,30,30
Accuracy: 97.52 %

Figure 20: Deep Neural Network probability plots varying from number of hidden layers

The above plots are the fitted predicted probability for each of three missile calibers, based on four hidden layers with different number of hidden units. Based on the time of correctly classifying and its accuracy distribution, 4 hidden layers looks do a better job.

However, with these four hidden layers containing 30 hidden units, we still cannot get 100% correct classification in the first 5 seconds. Then we transform the data inputs and put it into the new deep feed-forward model again.

3.3.3 Deep Feed-forward Neural Network Analysis Modeling

For deep feed-forward neural network modeling, I use “keras” R package and convert x, y, z and time into a long multivariate vector.

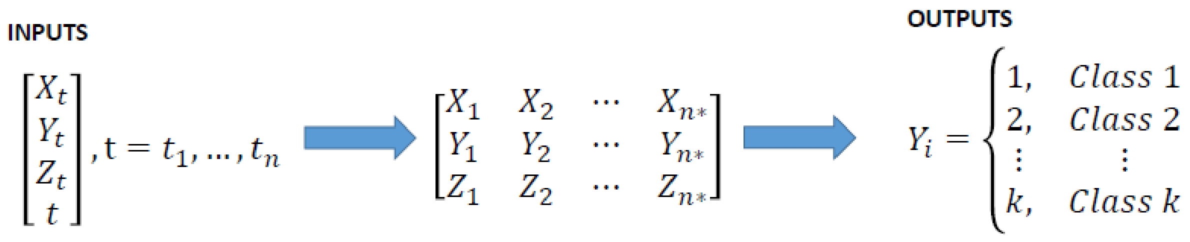

INPUTS
$\begin{bmatrix} X_t \\ Y_t \\ Z_t \\ t \end{bmatrix}, t = t_1, \dots, t_n$
$\begin{bmatrix} X_1 & X_2 & \cdots & X_{n*} \\ Y_1 & Y_2 & \cdots & Y_{n*} \\ Z_1 & Z_2 & \cdots & Z_{n*} \end{bmatrix}$
OUTPUTS
$Y_i = \begin{cases} 1, & Class\ 1 \\ 2, & Class\ 2 \\ \vdots & \vdots \\ k, & Class\ k \end{cases}$

Figure 21: Multivariate Missile Classification

In this way, if we subset the data into 2000 ms (two seconds), there would be 40 time points (2*20) totally in the subset data set, so the input becomes :

$$(\vec{X}_2', \vec{Y}_2', \vec{Z}_2') = (X_1, X_2, \cdots, X_{40}, Y_1, Y_2, \cdots, Y_{40}, Z_1, Z_2, \cdots, Z_{40}).$$

For the training dataset, I subset the dataset with the first 1000 ms, and put the input vector of length 60 (20 Xs, 20Ys, and 20Zs) in the input layer into the network. With 4 hidden layers, containing 25 units each, the network becomes complex, but is able to produce 100% correct classification for the independent Testing data.

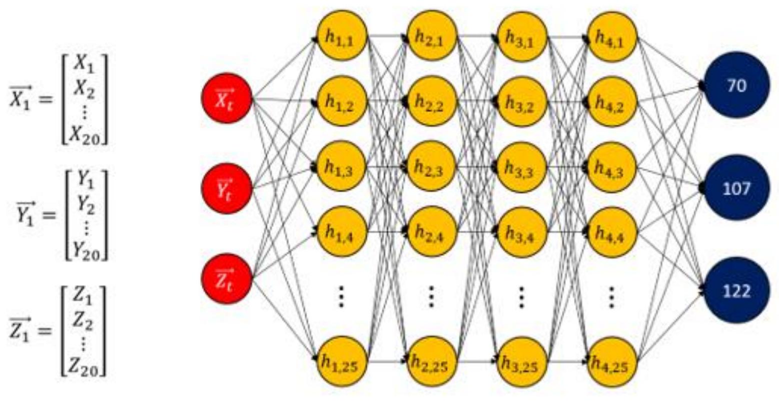
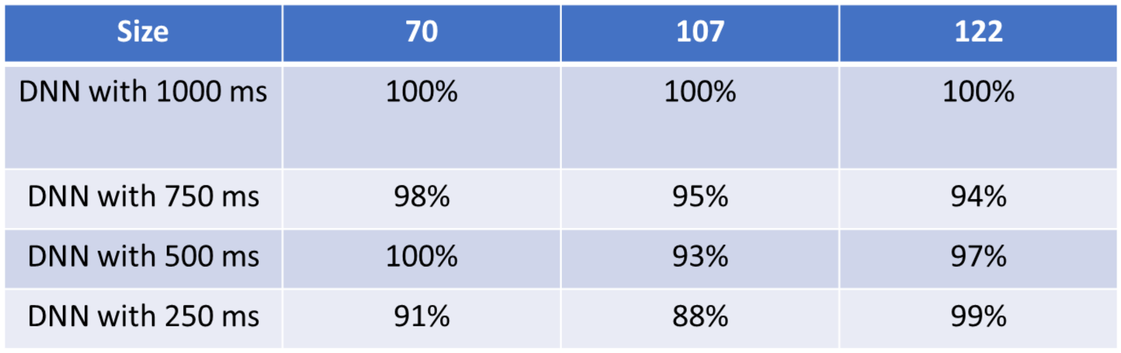

$\vec{X_1} = \begin{bmatrix} X_1 \\ X_2 \\ \vdots \\ X_{20} \end{bmatrix}$
$\vec{Y_1} = \begin{bmatrix} Y_1 \\ Y_2 \\ \vdots \\ Y_{20} \end{bmatrix}$
$\vec{Z_1} = \begin{bmatrix} Z_1 \\ Z_2 \\ \vdots \\ Z_{20} \end{bmatrix}$
$\vec{X_t}$
$\vec{Y_t}$
$\vec{Z_t}$
$h_{1,1}$ $h_{1,2}$ $h_{1,3}$ $h_{1,4}$ $h_{1,25}$
$h_{2,1}$ $h_{2,2}$ $h_{2,3}$ $h_{2,4}$ $h_{2,25}$
$h_{3,1}$ $h_{3,2}$ $h_{3,3}$ $h_{3,4}$ $h_{3,25}$
$h_{4,1}$ $h_{4,2}$ $h_{4,3}$ $h_{4,4}$ $h_{4,25}$
70
107
122

Figure 22: Rapid Missile / Rocket Classification Process

After we run the above deep feed-forward neural network modeling with the updated transformed inputs, our prediction accuracy tables are as below:



Size	70	107	122
DNN with 1000 ms	100%	100%	100%
DNN with 750 ms	98%	95%	94%
DNN with 500 ms	100%	93%	97%
DNN with 250 ms	91%	88%	99%

Figure 23: Rapid Missile / Rocket Classification Accuracy Table

From the above table, we can see with the 1000 ms neural network, we can classify three missiles 100% correctly. And with the time increasing, the accuracy is also developing.

Chapter 4

Conclusion and Discussion

In this thesis, we presented two methods to obtain the correct classification for the missile radar data in the first 5 seconds. In chapter 2, we performed regular neural network (one-hidden-layer neural network) to classify the three missiles. We compared the accuracy varying from different hidden units and pick the optimal hidden units for the final basic neural network model. In the test dataset, we proved the we can classify three rockets in the first 5 seconds around 98% correctly. In chapter 3, we performed deep neural network with and without transformed inputs (x, y, z and time). For the untransformed dataset, we used the "keras" R package to construct deep feed-forward neural network models for different hidden layers. By experimenting on different hidden layers and units, we finally pick the optimal one. For the transformed dataset which we transformed the four inputs into three multivariate input vectors, after trying different number of hidden layers and units based on the thumb rule, we found out the deep feed-forward neural network with 4 hidden layers each containing 25 units is able to provide 100% classification in the 1.25 seconds. In a word, we have shown that advanced machine learning techniques, like deep learning (Deep Neural Network) can perform very well when it comes to the classification problem for the complex data.

Future research will focus on expanding the design space to cover more general collections of missiles, the use of higher fidelity data. And develop recurrent neural network to treat the inputs as a times series data and add the memory ability into the neural network so that it can classify the objects in a shorter time with 100% accuracy. The other aspect of future research will be working on if there is a possibility that we can combine the recurrent neural network (RNN) with deep feedforward neural network (DNN) as RNN is good for time series prediction and DNN is good for pattern recognition.

In addition, in our paper, we don't have the issue of overfitting because the data we have is stimulated and we already split the data into training, validation and testing datasets. Therefore, we don't use the technique Dropout to solve the overfitting problem which neural networks usually bring out.

Finally, based on our deep feed-forward neural network model, it can provide a statistical approach for missile engineers to get the high-efficiency classification in a relatively short time given the first 1-2 second data sets.

References

1. Carpenter, M., Hartfield, R., and Zhou, L. (2018), “Deep Learning Neural Networks for Trajectory Analysis”, accepted, AIAA SciTech 2019, San Diego, CA.
2. Carpenter, M., Hartfield, R., and Ahuja, V.(2017), “Surrogate Modeling for Surface Vorticity”, Proceedings of AIAA SciTech 2017, Dallas, TX, January 2017.
3. Albarado, K., Hartfield, R., Carpenter, M., Burkhalter, and Ritz, S. (2012), “Rapid Missile Classification for Early Launch Using Neural Networks”, Proceedings AIAA 10th Annual U.S. Missile Defense Conference and Exhibit, Washington, DC, March 26-28, 2012, 16 pages. Classified Conference.
4. Hartfield, R., Carpenter, M., Randall, W., and Hundley, J. (2012) “Unmanned Air Vehicles (UAV): Safety Event Prediction, and Classification”, AIAA-2012-5427, Proceedings of the 12th AIAA Aviation Technology, Integration, and Operations (ATIO) Conference, Indianapolis, IN, September 17-19, 2012, 433-457.
5. Carpenter, M., Hartfield, R., and Burkhalter, J., “A Comprehensive Approach to Cataloging Missile Aerodynamic Performance using Surrogate Modeling Techniques and Statistical Learning”, Proceedings of the 29th AIAA Applied Aerodynamics Conference, 27-30, June 2011, Honolulu, Hawaii.
6. Hastie, T., Tibshirani, R., and Friedman, J., The Elements of Statistical Learning: Data Mining, Inference, and Prediction, Second Edition, Springer-Verlag, 2008

7. Gunther, Frauke and Fritsch, Stefan, “neuralnet: Training of Neural Networks,” The R Journal, V.2., pp.30-38, 2010
8. Mark Carpenter, Norman Speakman, and Roy J. Hartfield. "Rapid Characterization of Munitions Using Neural Networks", AIAA Atmospheric Flight Mechanics Conference, AIAA SciTech Forum, (AIAA 2016-0787)
9. Y, Guo, Y, Liu, A, Oerlemans. S, Lao., S. Wu, MS. Lew - Neurocomputing, 2016 – Elsevier
10. Bart van Merriënboer, Dzmitry Bahdanau, Vincent Dumoulin, Dmitriy Serdyuk, David Warde-Farley, Jan Chorowski, Yoshua Bengio, The Blocks and Fuel: Frameworks for deep learning, arXiv:1506.00619 ,2015
11. Jurgen Schmidhuber, Deep learning in neural networks: an overview, Neural Networks, Elsevier, 2015
12. Anup Badhe, Using Deep Learning Neural Networks to find best performing audience segments, International Journal of Scientific and Technology research, Vol 5, Issue 04, April 2016, ISSN 2277-8616
13. Geoffrey Hinton, Li Deng, Dong Yu, George E. Dahl, Abdel-rahman Mohamed, Navdeep Jailty, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen , Tara N. Sainath, and Brian Kingsbury, Deep Neural Networks for Acoustic Modeling in Speech Recognition, IEEE Signal Processing Magazine, Nov 2012
14. Alexander Toshev, Christian Szegedy, DeepPose: Human Pose estimation via deep neural networks, CVPR2014
15. Dan Ciresan, Ueli Mecier and Jurgen Schmidhuber, Multi-column Deep Neural Networks for Image Classfication, IEEE, 2012

16. Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, Ruslan Salakhutdinov, Dropout: A Simple Way to Prevent Neural Networks from Overfitting, Journal of Machine Learning Research 15 (2014) 1929-1958
17. Yong Xu, Jun Du, Li-Rong Dai, Chin-Hui Lee, Fellow, IEEE, A Regression Approach to Speech Enhancement Based on Deep Neural Netoworks, ACM transactions on Audio , Speech and Language processing, Vol. 23, No.1, Jan 2015
18. Ilya Sutskever, James Martens, George Dahl, Geoffrey Hinton, On the importance of initialization and momentum in deep learning, Proceedings of the 30th international conference on machine learning, 2013
19. Libo Zhang, Tiejian Luo, Fei Zhang, Yanjun Wu, A Recommendation Model Based on Deep Neural Network, IEEE Access, 2018
20. M. C. Limas, E. P.V. G. Joaquin B. Ordieres Mere, F. J. M. de pison Ascacibar, A. V. P. Espinoza, and F. A. Elias. AMORE Flexible Neural Network Package, 2007. URL http://wiki.r-project.org/rwiki/doku.php?id=packages:cran:amore. R package version 0.2-1
21. Venables, W. N. and Ripley, B. D., Modern Applied Statistics wish S. Fouth edition, Springer, 2002
22. Ripley, B. D., Pattern Recognition and Neural Networks. Cambridge 1996
23. LeCun, Y., et al. (1990). Handwritten digit recognition with a back-propagation network. In Advances in neural information processing systems (pp. 396-404).
24. Robert Hecht-Nielsen, III.3- Theory of the Backpropagation Neural Network. Neural Networks of Perception 1992, p65-93.

25. Ian Goodfellow, Yoshua Bengio, Aaron Courville, Deep Learning, 2016 Massachusetts Institute of Technology.

26. H.Gish, A probabilistic approach to the understanding and training of neural network classifiers, Proc. IEEE Int. Conf. Acoustic, Speech, Signal Processing, 1990, pp. 1361-1364

27. M.D. Richard and R. Lippmann, Neural Network classifiers estimate Bayesian a posteriori probabilities, Neural Comput, vol.3, pp. 461-483, 1991.

28. Jean M. Steppe, Kenneth W. Bauer Jr., Improved feature screening in feedforward neural networks, Neurocomput., Vol. 13, pp. 47-58, 1996

29. Geoffrey Hintonl, et.al , Deep Neural Networks for Acoustic Modeling in Speech Recognition: The shared views of four research groups, IEEE Signal Processing Magazine, Vol. 29, Issue 6.

30. Bengio, Y. (2009), Learning deep architectures for AI. Foundations and Trends in Machine Learning, 2, 1-127.

31. Vincent, P., Larochelle, H., Bengio, Y., & Manzagol, P.-A. (2008), Extracting and composing robust features with denoising autoencoders. ICML 2008.

32. Hinton, G. E., Osindero, S., & Teh, Y. (2006). A fast learning algorithm for Deep belief nets. Neural Computation, 18, 1527-1554.