



UNIVERSITEIT VAN PRETORIA  
UNIVERSITY OF PRETORIA  
YUNIBESITHI YA PRETORIA

FACULTY OF ENGINEERING, BUILT ENVIRONMENT AND INFORMATION  
TECHNOLOGY

DEPARTMENT OF ELECTRICAL, ELECTRONIC AND COMPUTER  
ENGINEERING  
<http://www.up.ac.za/eece>

## EAI732: Intelligent Systems

### Assignment 2: Neural Networks and Optimisation

*Compiled By*  
**Sholto Armstrong**  
14036071

15<sup>th</sup> April, 2018

## **DECLARATION OF ORIGINALITY**

### **UNIVERSITY OF PRETORIA**

The University of Pretoria places great emphasis upon integrity and ethical conduct in the preparation of all written work submitted for academic evaluation.

While academic staff teach you about referencing techniques and how to avoid plagiarism, you too have a responsibility in this regard. If you are at any stage uncertain as to what is required, you should speak to your lecturer before any written work is submitted.

You are guilty of plagiarism if you copy something from another author's work (e.g. a book, an article or a website) without acknowledging the source and pass it off as your own. In effect you are stealing something that belongs to someone else. This is not only the case when you copy work word-for-word (verbatim), but also when you submit someone else's work in a slightly altered form (paraphrase) or use a line of argument without acknowledging it. You are not allowed to use work previously produced by another student. You are also not allowed to let anybody copy your work with the intention of passing it off as his/her work.

Students who commit plagiarism will not be given any credit for plagiarised work. The matter may also be referred to the Disciplinary Committee (Students) for a ruling. Plagiarism is regarded as a serious contravention of the University's rules and can lead to expulsion from the University.

The declaration which follows must accompany all written work submitted while you are a student of the University of Pretoria. No written work will be accepted unless the declaration has been completed and attached.

Full names of student: Sholto Armstrong

Student number: 14036071

Topic of work: Assignment 1: Probability Theory

#### **Declaration**

1. I understand what plagiarism is and am aware of the University's policy in this regard.
2. I declare that this assignment report is my own original work. Where other people's work has been used (either from a printed source, Internet or any other source), this has been properly acknowledged and referenced in accordance with departmental requirements.
3. I have not used work previously produced by another student or any other person to hand in as my own.
4. I have not allowed, and will not allow, anyone to copy my work with the intention of passing it off as his or her own work.

SIGNATURE: \_\_\_\_\_ DATE: 7 April 2018

# TABLE OF CONTENTS

<b>1</b>	<b>Introduction</b>	<b>2</b>
<b>2</b>	<b>Feed-Forward Neural Networks</b>	<b>2</b>
<b>3</b>	<b>Activation function</b>	<b>4</b>
<b>4</b>	<b>Backward Propagation</b>	<b>6</b>
<b>5</b>	<b>RPROP</b>	<b>9</b>
5.1	RPROP- . . . . .	9
5.2	RPROP+ . . . . .	9
5.3	iRPROP+ . . . . .	10
5.4	iRPROP- . . . . .	11
<b>6</b>	<b>Quickprop</b>	<b>11</b>
<b>7</b>	<b>Adaptations to Back-propagation</b>	<b>12</b>
7.1	ADAGRAD . . . . .	12
7.2	Momentum . . . . .	13
<b>8</b>	<b>Genetic Algorithm</b>	<b>14</b>
8.1	Selection . . . . .	14
8.2	Crossover . . . . .	14
8.3	Mutation . . . . .	14
<b>9</b>	<b>Particle Swarm Optimisation</b>	<b>15</b>
<b>10</b>	<b>Proben1</b>	<b>15</b>
<b>11</b>	<b>Evaluation Procedure</b>	<b>16</b>
11.1	Experiment 1: Testing with Proben1 . . . . .	18
11.2	Experiment 2: MNIST . . . . .	19
<b>12</b>	<b>Results</b>	<b>19</b>
12.1	Experiment 1: Testing with Proben1 . . . . .	19
12.2	Experiment 2: MNIST . . . . .	56
<b>13</b>	<b>Discussion</b>	<b>57</b>
<b>14</b>	<b>Summary</b>	<b>58</b>
<b>15</b>	<b>Conclusion</b>	<b>58</b>
<b>16</b>	<b>References</b>	<b>59</b>
16.1	cancer . . . . .	60
16.2	card . . . . .	107
16.3	flare . . . . .	155
16.4	gene . . . . .	179
16.5	horse . . . . .	251
16.6	heartc . . . . .	299

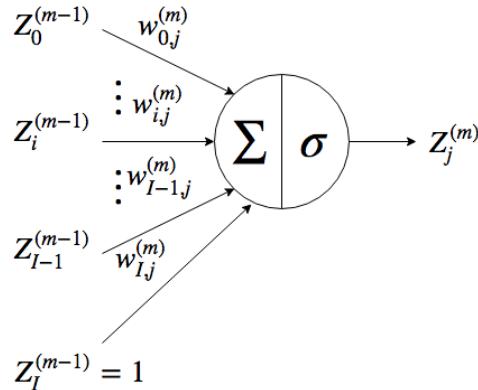
# 1 Introduction

Neural networks has been growing in popularity in recent years. This is partly due to the fact that it attempts to mimic the processes that happen in the biological brain. However, there are many differences between artificial neural networks and biological neural networks. However, artificial neural networks are a powerful tool which can be applied to a vast variety of problems from classification to regression. Due to this, it is of interest to explore neural networks and their inner-workings.

The aim of this report is to construct a basic fully connected feed forward neural network from first principles. The backward propagation algorithm is then to be developed for this network. After the generic backward-propagation is developed and tested, we shall explore other methods which can be used to update neural networks. These are to be discussed and compared with the normal backward propagation algorithm. More general optimisation techniques, which do not rely on gradient information, are also to be implemented. These include the genetic algorithm and the particle swarm optimisation algorithm. Finally, one of these techniques are used to train the neural network on the MNIST[1]. The results of this are then compared with a naive Bayesian classifier which was previously implemented, in order to determine the effectiveness of a neural network.

## 2 Feed-Forward Neural Networks

Neural networks are a powerful tool in machine learning which can be applied to a variety of problems and often provide excellent results. As the name implies, neural networks are roughly based on biological neurons. However, there are many differences between biological neurons and artificial neurons. The commonly established model for an artificial neuron is given in Fig. 1.



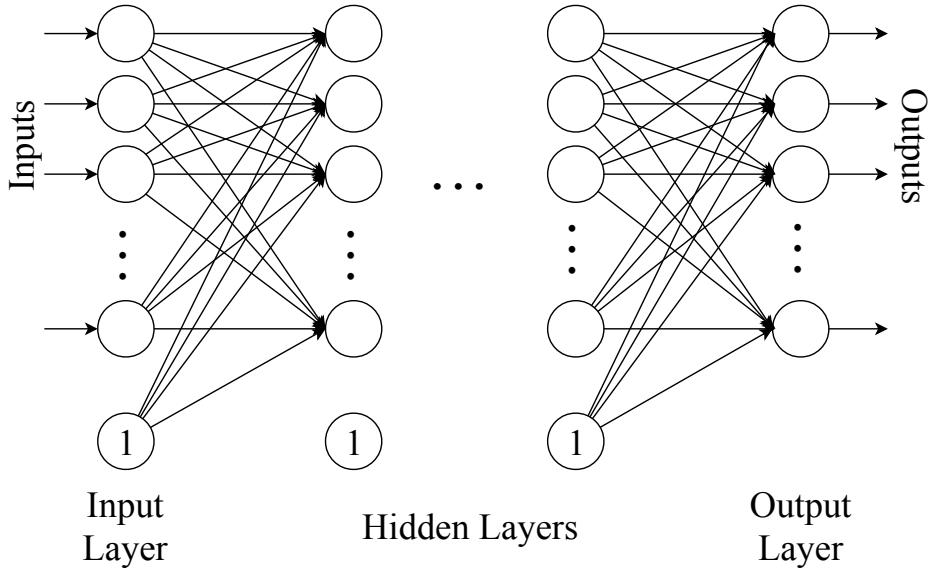
**Figure 1.** Diagram depicting a single neuron.

In this model, a weighted sum of all the inputs is calculated. An activation function  $\sigma$  is then applied to this sum in order to obtain the output of this node. We have also defined the last input as unity. This is known as a bias parameter. This gives the neuron a fixed point, around which, it can adapt the other parameters, as opposed to relating parameters to each-other. This, more often than not, leads to more accurate neural networks, which are faster to train. This is shown in Eqn. 2.1. Here we have defined  $m$  as the layer that the neuron belongs to,  $i$  as the index of the input and  $j$  as the index of the output. These relate to the position of this neuron in the neural network, which is described later in this section. Furthermore, It is important to note that the  $m$  in the superscript of the various parameters has been put in brackets. This has been done to assert that the parameter is associated to the  $m$ 'th layer as opposed

to taking the  $m$ 'th power.

$$Z_j^{(m)} = \sigma \left( \sum_{i=0}^I w_{i,j}^{(m)} Z_i^{(m-1)} \right) \quad (2.1)$$

These neurons can be aligned in various configurations. One such configuration is known as the fully-connected feed-forward neural network, shown in Fig. 2. This is one of the simplest configurations, but allows for quick and easy evaluation through matrix operations. Neural networks are commonly divided into 3 layers. This includes an input layer, which does not have an activation function, but rather connects the inputs to the relevant nodes in the next layer. After the input layers comes the hidden layers. These are comprised of fully connected neurons shown in Fig. 1. These are connected to a final layer, known as the output layer.



**Figure 2.** Topology diagram of a fully-connected feed-forward neural network.

We can now use Eqn. 2.1 to derive the outputs for this network. The output of a layer is shown in Eqn. 2.2. This can then be used to calculate the output of the network by setting the input to  $\mathbf{Z}^{(0)}$  and iteratively applying this formula until the last layer is reached. This process is known as forward-propagation.

$$\mathbf{Z}^{(m)} = \sigma \left( \mathbf{W}^{(m)} \cdot \mathbf{Z}^{(m-1)} \right) \quad (2.2)$$

Here we have defined the weight matrix,  $\mathbf{W}^{(m)}$ , as shown in Eqn. 2.3 and the output/input vector,  $\mathbf{Z}^{(m)}$ , as shown in Eqn. 2.4. Finally, pseudocode for forward propagation is shown in Alg. 1.

$$\mathbf{W}^{(m)} = \begin{pmatrix} w_{0,0}^{(m)} & w_{1,0}^{(m)} & w_{2,0}^{(m)} & \dots & w_{I,0}^{(m)} \\ w_{0,1}^{(m)} & w_{1,1}^{(m)} & w_{2,1}^{(m)} & \dots & w_{I,1}^{(m)} \\ w_{0,2}^{(m)} & w_{1,2}^{(m)} & w_{2,2}^{(m)} & \dots & w_{I,2}^{(m)} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ w_{0,J}^{(m)} & w_{1,J}^{(m)} & w_{2,J}^{(m)} & \dots & w_{I,J}^{(m)} \end{pmatrix} \quad (2.3)$$

$$\mathbf{Z}^{(m)} = \begin{pmatrix} Z_0^{(m)} \\ Z_1^{(m)} \\ \vdots \\ Z_J^{(m)} \end{pmatrix} \quad (2.4)$$

---

**Algorithm 1** Forward Propagation

---

```

1: procedure FORWARD-PROPAGATION(input Vector)
2:   Layers  $\leftarrow$  An M dimentional array containing the weight matrix,  $\mathbf{W}^{(m)}$ , for each layer
3:   Output Vector  $\leftarrow$  input Vector
4:   for m from 1 to M do
5:     Output Vector  $\leftarrow$  Layers[m].WeightMat  $\cdot$  Output Vector
6:   end for
7: return Output Vector
8: end procedure

```

---

### 3 Activation function

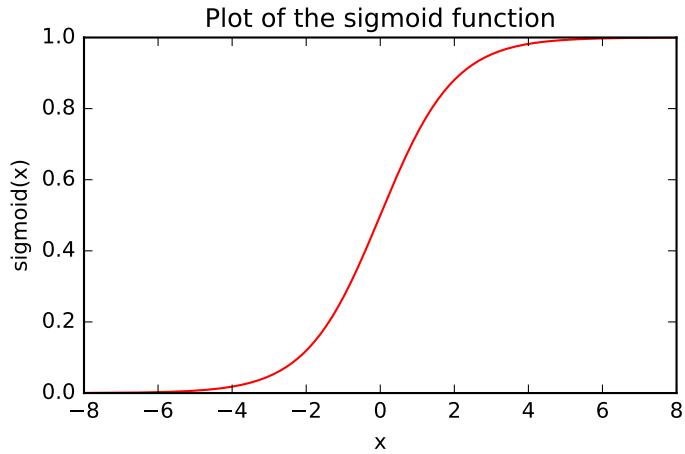
In section 2, the general model of a basic feed-forward neural network has been discussed. One important aspect of this model is known as the activation function,  $\sigma$ , shown in Fig. 1. Thus, it is important to discuss the selection of the activation function. The most simple of which, is known as the unity activation function. This is when no activation function is used at all, which allows for a very simplistic model. However, this is unable to model non-linearities, reducing the complexity of the function that the neural network is able to model. This is due to the fact that any combination of linear functions is still linear.

When using the neural network for classification, one would consider using a step-function as the activation function. However, this function is not continuous and hence contains singularities in the derivative, which provides significant advantages when attempting to train the network. Therefore, it is sensible to consider the sigmoid function, which is similar to the step function. It, however, is continuous and has a compact derivative. Apart from being continuous, the sigmoid function is also monotonically increasing, non-constant and bounded. These are all the criteria required for the Universal Approximation theorem[2] to be applicable. This states that a neural network can approximate any function to an arbitrary accuracy. Due to this, a sigmoid function is a promising choice for an activation function and will be used throughout this paper. The sigmoid function is given in Eqn. 3.1 and depicted in Fig. 3.

$$\sigma(x) = \frac{1}{1 + e^{-x}} \quad (3.1)$$

The derivative of the sigmoid function is calculated in Eqn. 3.2.

$$\begin{aligned}
\frac{\partial}{\partial x} \sigma(x) &= \frac{\partial}{\partial x} \frac{1}{1+e^{-x}} \\
&= \frac{\left(\frac{\partial}{\partial x} 1\right)(1+e^{-x}) - \frac{\partial}{\partial x}(1+e^{-x})}{(1+e^{-x})^2} \\
&= \frac{e^{-x} + 1 - 1}{(1+e^{-x})^2} \\
&= \frac{1+e^{-x}}{(1+e^{-x})^2} - \left(\frac{1}{(1+e^{-x})}\right)^2 \\
&= \sigma(x)(1-\sigma(x))
\end{aligned} \tag{3.2}$$



**Figure 3.** Graph of the sigmoid function

An alternative function to the sigmoid function, is the hyperbolic tan (tanh) function. The only difference between the tanh function and the sigmoid function, is the range of the outputs. The tanh function is bounded by  $-1$  and  $1$ , where the sigmoid function is bounded by  $0$  and  $1$ . Eqn. 3.3, defines the tanh function in terms of the sigmoid function. Since the sigmoid function can be interoperated directly as a likelihood. It is more convenient than a tanh function. However, this choice is completely arbitrary.

$$\tanh(x) = 2\sigma(2x) - 1 \tag{3.3}$$

The final function under consideration is the softmax function, which acts as a continuous max function. This function is useful when dealing with classification problems, where only one result can be correct (one hot encoding). A similar result can be obtained by selecting the most likely result from multiple sigmoidal outputs. Due to this, the sigmoid function can be applied to a broader range of problems. Therefore, the softmax function isn't explored in this report, even though it is a useful activation function.

## 4 Backward Propagation

Once the neural network model is designed, a method is required to train it. For this, a measure of the accuracy of the network is required. The L2 norm error is a common choice for this and is given in Eqn. 4.1, where  $t_{j,d}$  is the target value for the selected output and  $D$  is the size of the dataset.

$$E(w) = \frac{1}{2} \sum_{d=0}^D \sum_{j=0}^J \left( Z_{j,d}^{(M)} - t_{j,d} \right)^2 \quad (4.1)$$

It is now possible to apply gradient decent to optimise the parameters. In order to do this, we will first consider the update step for the output layer. This requires that the partial derivative of the error with respect to the relevant weights be calculated. For simplification, the error is calculated over a single item from the dataset. This can then be applied to each item in the dataset separately, or stored to update the weights in a batch process. Therefore, the partial derivatives can be calculated as follows:

$$\frac{\partial}{\partial w_{i,j}^{(M)}} E(w) = \frac{\partial}{\partial w_{i,j}^{(M)}} \frac{1}{2} \sum_{k=0}^J \left( Z_k^{(M)} - t_k \right)^2 \quad (4.2)$$

We now note that:

$$\frac{\partial}{\partial w_{i,j}^{(M)}} \left( Z_k^{(M)} - t_k \right)^2 = \begin{cases} \frac{\partial}{\partial w_{i,j}^{(M)}} \left( Z_j^{(M)} - t_j \right)^2, & k = i \\ 0, & k \neq i \end{cases} \quad (4.3)$$

From this, the partial derivative of the error with respect to the weights can be written as follows:

$$\begin{aligned} \frac{\partial}{\partial w_{i,j}^{(M)}} E(w) &= \frac{\partial}{\partial w_{i,j}^{(M)}} \frac{1}{2} \left( Z_j^{(M)} - t_j \right)^2 \\ &= \left( \frac{\partial}{\partial w_{i,j}^{(M)}} Z_j^{(M)} \right) \left( Z_j^{(M)} - t_j \right) \\ &= \left( \frac{\partial}{\partial w_{i,j}^{(M)}} \sigma(p_j^{(M)}) \right) \left( Z_j^{(M)} - t_j \right) \end{aligned} \quad (4.4)$$

Here  $p_j^{(m)}$  is defined as follows:

$$p_j^{(m)} = \sum_{i=0}^{I^{(m)}} w_{i,j}^{(m)} Z_i^{(m-1)} \quad (4.5)$$

Eqn. 3.2 is then substituted into Eqn. 4.4 to give yield Eqn. 4.6.

$$\frac{\partial}{\partial w_{i,j}^{(M)}} E(w) = \sigma(p_j^{(M)}) \left( 1 - \sigma(p_j^{(M)}) \right) \left( Z_j^{(M)} - t_j \right) \frac{\partial}{\partial w_{i,j}^{(M)}} \sum_{k=0}^I w_{k,j}^{(M)} Z_k^{(M-1)} \quad (4.6)$$

Where,

$$\frac{\partial}{\partial w_{i,j}^{(M)}} \sum_{k=0}^I w_{k,j}^{(M)} Z_k^{(M-1)} = \begin{cases} Z_k^{(M-1)}, & k = i \\ 0, & k \neq i \end{cases} \quad (4.7)$$

Substituting Eqn. 4.7 into Eqn. 4.6 we can obtain 4.8.

$$\begin{aligned} \frac{\partial}{\partial w_{i,j}^{(M)}} E(w) &= Z_j^{(M)} \left( 1 - Z_j^{(M)} \right) \left( Z_j^{(M)} - t_j \right) Z_i^{(M-1)} \\ &= \delta_j^{(M)} Z_i^{(M-1)} \end{aligned} \quad (4.8)$$

In Eqn. 4.8,  $\delta_j^{(M)}$  is used to clean up the notation. This is defined in Eqn. 4.9.

$$\delta_j^{(M)} = Z_j^{(M)} \left( 1 - Z_j^{(M)} \right) \left( Z_j^{(M)} - t_j \right) \quad (4.9)$$

The partial derivative of the error with respect to the weights in the hidden layers, can be calculated with the use of Eqn. 4.2. This is shown below.

$$\begin{aligned} \frac{\partial}{\partial w_{i,j}^{(m)}} E(w) &= \frac{\partial}{\partial w_{i,j}^{(m)}} \frac{1}{2} \sum_{k=0}^{J^{(M)}} \left( Z_k^{(M)} - t_k \right)^2 \\ &= \sum_{k=0}^{J^{(M)}} \left( Z_k^{(M)} - t_k \right) \frac{\partial}{\partial w_{i,j}^{(m)}} \sigma \left( p_k^{(M)} \right) \\ &= \sum_{k=0}^{J^{(M)}} \left( Z_k^{(M)} - t_k \right) \sigma \left( p_k^{(M)} \right) \left( 1 - \sigma \left( p_k^{(M)} \right) \right) \frac{\partial}{\partial w_{i,j}^{(m)}} p_k^{(M)} \\ &= \sum_{k=0}^{J^{(M)}} \delta_k^{(M)} \frac{\partial}{\partial w_{i,j}^{(m)}} p_k^{(M)} \end{aligned} \quad (4.10)$$

Note that the delta calculated in the output layer is used when calculating the partial derivative in the hidden layers. This is vital when calculating the partial derivative for any given hidden layer. It is now possible to continue with the evaluation of  $\frac{\partial}{\partial w_{i,j}^{(m)}} E(w)$ , given in Eqn. 4.10. This is done below.

$$\begin{aligned} \frac{\partial}{\partial w_{i,j}^{(m)}} E(w) &= \sum_{k=0}^{J^{(M)}} \delta_k^{(M)} \frac{\partial}{\partial w_{i,j}^{(m)}} \sum_{h=0}^{I^{(M)}} w_{h,k}^{(M)} Z_h^{(M-1)} \\ &= \sum_{k=0}^{J^{(M)}} \delta_k^{(M)} \sum_{h=0}^{I^{(M)}} w_{h,k}^{(M)} \frac{\partial}{\partial w_{i,j}^{(m)}} Z_h^{(M-1)} \\ &= \sum_{k=0}^{J^{(M)}} \sum_{h=0}^{I^{(M)}} \delta_k^{(M)} w_{h,k}^{(M)} \sigma \left( p_h^{(M-1)} \right) \left( 1 - \sigma \left( p_h^{(M-1)} \right) \right) \frac{\partial}{\partial w_{i,j}^{(m)}} p_h^{(M-1)} \\ &= \sum_{h=0}^{I^{(M-1)}} Z_h^{(M-1)} \left( 1 - Z_h^{(M-1)} \right) \left( \sum_{k=0}^{J^{(M)}} \delta_k^{(M)} w_{h,k}^{(M)} \right) \frac{\partial}{\partial w_{i,j}^{(m)}} p_h^{(M-1)} \\ &= \sum_{h=0}^{I^{(M-1)}} \delta_h^{(M-1)} \frac{\partial}{\partial w_{i,j}^{(m)}} p_h^{(M-1)} \end{aligned} \quad (4.11)$$

We have now defined delta as shown in Eqn. 4.12.

$$\delta_i^{(m)} = \begin{cases} Z_i^{(M)} \left( 1 - Z_i^{(M)} \right) \left( Z_i^{(M)} - t_i \right) & m = M \\ Z_i^{(m)} \left( 1 - Z_i^{(m)} \right) \left( \sum_{j=0}^{J^{(m+1)}} \delta_j^{(m+1)} w_{i,j}^{(m+1)} \right) & m < M \end{cases} \quad (4.12)$$

Comparing Eqn. 4.10 and Eqn. 4.11, we can now define an iterative function for the partial derivative.

This is done in Eqn. 4.13.

$$\begin{aligned}
\frac{\partial}{\partial w_{i,j}^{(m)}} E(w) &= \sum_{h=0}^{J^{(m+1)}} \delta_h^{(m+1)} \frac{\partial}{\partial w_{i,j}^{(m)}} p_h^{(m+1)} \\
&= \sum_{h=0}^{J^{(m+1)}} \delta_h^{(m+1)} \frac{\partial}{\partial w_{i,j}^{(m)}} \sum_{j=0}^{J^{(m)}} w_{j,h}^{(m+1)} Z_j^{(m)} \\
&= \sum_{h=0}^{J^{(m+1)}} \delta_h^{(m+1)} \frac{\partial}{\partial w_{i,j}^{(m)}} \sum_{g=0}^{J^{(m)}} w_{g,h}^{(m+1)} Z_g^{(m)} \\
&= \sum_{h=0}^{J^{(m+1)}} \delta_h^{(m+1)} w_{j,h}^{(m+1)} \frac{\partial}{\partial w_{i,j}^{(m)}} Z_j^{(m)} \\
&= \sum_{h=0}^{J^{(m+1)}} \delta_h^{(m+1)} w_{j,h}^{(m+1)} Z_j^{(m)} \left(1 - Z_j^{(m)}\right) Z_i^{(m-1)} \\
&= Z_j^{(m)} \left(1 - Z_j^{(m)}\right) Z_i^{(m-1)} \sum_{h=0}^{J^{(m+1)}} \delta_h^{(m+1)} w_{j,h}^{(m+1)} \\
&= \delta_j^{(m)} Z_i^{(m-1)}
\end{aligned} \tag{4.13}$$

When considering a fully-connected FFNN, these equations can be written in matrix form as shown in Eqn. 4.14.

$$\delta^{(m)} = \begin{cases} \mathbf{Z}^{(M)} \odot (\mathbf{1} - \mathbf{Z}^{(M)}) \odot (\mathbf{Z}^{(M)} - \mathbf{t}) & m = M \\ \mathbf{Z}^{(m)} \odot (\mathbf{1} - \mathbf{Z}^{(m)}) \odot (\mathbf{W}^{(m+1)} \cdot \delta^{(m+1)}) & m < M \end{cases} \tag{4.14}$$

The gradient decent step can then be applied as shown in Eqn. 4.15, where the learning rate,  $\eta$ , controls how far to move in the direction of the steepest downward gradient.

$$\mathbf{W}^{(m)}(t+1) = \mathbf{W}^{(m)}(t) - \eta \delta^{(m)} \cdot (\mathbf{Z}^{(m-1)})^T \tag{4.15}$$

Eqn. 4.15, assumes that a single input and target vector are considered at each time step. It is however possible to consider multiple inputs and targets in a single iteration. This is done by noting that each input contributes equity to the final error. Therefore, a summation over  $\delta$ 's can be performed before applying the update step. This allows batch updates to be performed. Both methods are implemented in a similar fashion. The pseudo-code for this is shown in Alg. 2.

---

### Algorithm 2 Back Propagation

---

```

1: procedure BACK-PROPAGATION(input Matrix, Target Matrix,  $\eta$ )
2:   Forward propagate each input and store  $Z^{(m-1)}$  and  $Z^{(m)}$  for each layer  $m$  where  $m \leftarrow 1$  to  $M$ 
3:    $errorContrib \leftarrow \sum_{inputs,targets} (\mathbf{Z}^{(M)} - \mathbf{t})$ 
4:   for  $m$  from  $M$  down to 1 do
5:     Calculate  $\delta^{(m)}$  using Eqn. 4.14.
6:     Update  $\mathbf{W}^{(m)}$  using Eqn. 4.15.
7:      $errorContrib \leftarrow (\mathbf{W}^{(m)} \cdot \delta^{(m)})$ 
8:   end for
9: end procedure

```

---

## 5 RPROP

Resilient back-propagation (RPROP) is a method which can be used to update the weights of a neural network. This algorithm increases the size of the steps taken if the direction of the steepest descent is consistent over consecutive iterations. This is due to the fact that a minima must still be ahead of the current position of a weight. Once the direction of the steepest decent changes, one can assume that the minima was passed as it now lies behind the position of the weight. Due to this, the step size must be decreased in order to descend into the minima, and not jump back to the previous value. Since this algorithm uses only the direction of the steepest descent and not the magnitude, it is resistant to an effect known as "The Vanishing Gradient Problem"[3]. This is caused by gradients which are less than one. As the error gets propagated backwards, these gradients multiply, which causes them to decrease until their effect is negligible. This becomes an issue when training deep neural networks or recurrent neural networks. Due to this, RPROP has a definite advantage over many other algorithms[4]. There are four main variants of RPROP described in [5]. These are discussed further in this chapter.

### 5.1 RPROP-

RPROP without weight-backtracking (RPROP-), is the simplest of all the RPROP algorithms. In order to implement this algorithm, we first set an initial  $\Delta_{i,j}^{(m)}$  for each weight in the neural network. We also define two learning rate adjustment parameters  $\eta^+$  and  $\eta^-$ , where  $0 < \eta^- < 1 < \eta^+$ . Finally, a maximum delta  $\Delta_{max}$  and minimum delta  $\Delta_{min}$  is defined. To apply the update step, we first check if the previous gradient multiplied with the current gradient is positive. This indicates that the minimum is still in the same direction as it was previously. Thus, we increase the previous delta  $\Delta_{i,j}^{(m)}(t-1)$  by a factor of  $\eta^+$ , ensuring that it does not exceed  $\Delta_{max}$ . When the current gradient is in the opposite direction of the previous gradient  $\frac{\partial E}{\partial w_{i,j}^{(m)}}(t) \cdot \frac{\partial E}{\partial w_{i,j}^{(m)}}(t-1) < 0$ , we know that we have passed a local minimum. Therefore, the learning rate is decreased by a factor of  $\eta^-$ , ensuring that the delta is no less than  $\Delta_{min}$ . Finally, no change is made to the delta if either gradient is 0. The delta can then be added to the current weights in the direction of the steepest decent. This process is summarised in Alg. 3.

---

#### Algorithm 3 RPROP-

---

```

1: procedure RPROP-( $w_{ij}^{(m)}$ ,  $\Delta_{i,j}^{(m)}(t-1)$ ,  $\frac{\partial E}{\partial w_{i,j}^{(m)}}(t)$ ,  $\frac{\partial E}{\partial w_{i,j}^{(m)}}(t-1)$ ,  $\eta^+$ ,  $\eta^-$ ,  $\Delta_{min}$ ,  $\Delta_{max}$ )
2:    $hold \leftarrow \frac{\partial E}{\partial w_{i,j}^{(m)}}(t) \cdot \frac{\partial E}{\partial w_{i,j}^{(m)}}(t-1)$ 
3:    $\Delta_{i,j}^{(m)}(t) \leftarrow \begin{cases} \min(\Delta_{i,j}^{(m)}(t-1) \cdot \eta^+, \Delta_{max}) & hold > 0 \\ \max(\Delta_{i,j}^{(m)}(t-1) \cdot \eta^-, \Delta_{min}) & hold < 0 \\ \Delta_{i,j}^{(m)}(t-1) & hold = 0 \end{cases}$ 
4:    $w_{ij}^{(m)} \leftarrow w_{ij}^{(m)} - \text{sign}\left(\frac{\partial E}{\partial w_{i,j}^{(m)}}(t)\right) \cdot \Delta_{i,j}^{(m)}(t)$ 
5: end procedure

```

---

### 5.2 RPROP+

RPROP with weight-backtracking[6, 7, 8] follows similar logic to RPROP without weight-backtracking. The only difference is that when the gradient changes direction, RPROP+ reverts the weight back to its previous location. This is shown in Alg. 4.

---

**Algorithm 4** RPROP+

---

- 1: **procedure** RPROP+( $w_{ij}^{(m)}$ ,  $\Delta_{i,j}^{(m)}(t-1)$ ,  $\frac{\partial E}{\partial w_{i,j}^{(m)}}(t)$ ,  $\frac{\partial E}{\partial w_{i,j}^{(m)}}(t-1)$ ,  $\eta^+$ ,  $\eta^-$ ,  $\Delta_{min}$ ,  $\Delta_{max}$ )
- 2:      $hold \leftarrow \frac{\partial E}{\partial w_{i,j}^{(m)}}(t) \cdot \frac{\partial E}{\partial w_{i,j}^{(m)}}(t-1)$
- 3:      $\Delta_{i,j}^{(m)}(t) \leftarrow \begin{cases} \min(\Delta_{i,j}^{(m)}(t-1) \cdot \eta^+, \Delta_{max}) & hold > 0 \\ \max(\Delta_{i,j}^{(m)}(t-1) \cdot \eta^-, \Delta_{min}) & hold < 0 \\ \Delta_{i,j}^{(m)}(t-1) & hold = 0 \end{cases}$
- 4:      $\frac{\partial E}{\partial w_{i,j}^{(m)}}(t-1) \leftarrow \begin{cases} 0 & hold < 0 \\ \frac{\partial E}{\partial w_{i,j}^{(m)}}(t) & hold \geq 0 \end{cases}$
- 5:      $\Delta w_{ij}^{(m)}(t) \leftarrow \begin{cases} \text{sign}\left(\frac{\partial E}{\partial w_{i,j}^{(m)}}(t-1)\right) \cdot \Delta_{i,j}^{(m)}(t) & hold \neq 0 \\ \Delta w_{ij}^{(m)}(t-1) & hold = 0 \end{cases}$
- 6:      $w_{ij}^{(m)} \leftarrow w_{ij}^{(m)} - \Delta w_{ij}^{(m)}(t)$
- 7: **end procedure**

---

### 5.3 iRPROP+

Improved RPROP+ attempts to avoid backtracking if the overall error of the algorithm has decreased after taking a step. This reduces unnecessary iterations where weights revert to their previous positions. Alg. 5 shows how this is implemented.

---

**Algorithm 5** iRPROP+

---

- 1: **procedure** iRPROP+( $w_{ij}^{(m)}$ ,  $\Delta_{i,j}^{(m)}(t-1)$ ,  $\frac{\partial E}{\partial w_{i,j}^{(m)}}(t)$ ,  $\frac{\partial E}{\partial w_{i,j}^{(m)}}(t-1)$ ,  $\eta^+$ ,  $\eta^-$ ,  $\Delta_{min}$ ,  $\Delta_{max}$ ,  $E(t)$ ,  $E(t-1)$ )
- 2:      $hold \leftarrow \frac{\partial E}{\partial w_{i,j}^{(m)}}(t) \cdot \frac{\partial E}{\partial w_{i,j}^{(m)}}(t-1)$
- 3:      $\Delta_{i,j}^{(m)}(t) \leftarrow \begin{cases} \min(\Delta_{i,j}^{(m)}(t-1) \cdot \eta^+, \Delta_{max}) & hold > 0 \\ \max(\Delta_{i,j}^{(m)}(t-1) \cdot \eta^-, \Delta_{min}) & hold < 0 \\ \Delta_{i,j}^{(m)}(t-1) & hold = 0 \end{cases}$
- 4:      $\frac{\partial E}{\partial w_{i,j}^{(m)}}(t-1) \leftarrow \begin{cases} 0 & hold < 0 \\ \frac{\partial E}{\partial w_{i,j}^{(m)}}(t) & hold \geq 0 \end{cases}$
- 5:      $\Delta w_{ij}^{(m)}(t) \leftarrow \begin{cases} \text{sign}\left(\frac{\partial E}{\partial w_{i,j}^{(m)}}(t-1)\right) \cdot \Delta_{i,j}^{(m)}(t) & hold \neq 0 \text{ and } E(t) > E(t-1) \\ 0 & hold \neq 0 \text{ and } E(t) \leq E(t-1) \\ \Delta w_{ij}^{(m)}(t-1) & hold = 0 \end{cases}$
- 6:      $w_{ij}^{(m)} \leftarrow w_{ij}^{(m)} - \Delta w_{ij}^{(m)}(t)$
- 7: **end procedure**

---

## 5.4 iRPROP-

iRPROP- implements iRPROP+ without backtracking. For this, the gradient with relation to the error is set to zero when the direction of the gradient changes between iterations. This forces the algorithm to wait until the next iteration to update the relevant weight.

---

### Algorithm 6 iRPROP-

---

```

1: procedure iRPROP-( $w_{ij}^{(m)}$ ,  $\Delta_{i,j}^{(m)}(t-1)$ ,  $\frac{\partial E}{\partial w_{i,j}^{(m)}}(t)$ ,  $\frac{\partial E}{\partial w_{i,j}^{(m)}}(t-1)$ ,  $\eta^+$ ,  $\eta^-$ ,  $\Delta_{min}$ ,  $\Delta_{max}$ )
2:    $hold \leftarrow \frac{\partial E}{\partial w_{i,j}^{(m)}}(t) \cdot \frac{\partial E}{\partial w_{i,j}^{(m)}}(t-1)$ 
3:    $\Delta_{i,j}^{(m)}(t) \leftarrow \begin{cases} \min(\Delta_{i,j}^{(m)}(t-1) \cdot \eta^+, \Delta_{max}) & hold > 0 \\ \max(\Delta_{i,j}^{(m)}(t-1) \cdot \eta^-, \Delta_{min}) & hold < 0 \\ \Delta_{i,j}^{(m)}(t-1) & hold = 0 \end{cases}$ 
4:    $\frac{\partial E}{\partial w_{i,j}^{(m)}}(t-1) \leftarrow \begin{cases} 0 & hold < 0 \\ \frac{\partial E}{\partial w_{i,j}^{(m)}}(t) & hold \geq 0 \end{cases}$ 
5:    $w_{ij}^{(m)} \leftarrow w_{ij}^{(m)} - \text{sign}\left(\frac{\partial E}{\partial w_{i,j}^{(m)}}(t-1)\right) \cdot \Delta_{i,j}^{(m)}(t)$ 
6: end procedure

```

---

## 6 Quickprop

Quick propagation[9] attempts to speed up back propagation by estimating an optimal value for the learning rate. This is done by estimating the second derivative of the error with respect to the weight. This information is then used to set the learning rate. This can be done by assuming that the error vs. weight curve takes a positive parabolic form. Furthermore, we must assume that adjusting one weight does not drastically influence the error contributed by the other weights. Making these assumptions, we can approximate the error corresponding to each weight as shown in Eqn. 6.1. Its derivatives are then shown in Eqns. 6.2, 6.3 and 6.4.

$$E(w, t) = aw^2(t) + bw(t) + c \quad (6.1)$$

$$\frac{\partial E(w, t)}{\partial w} = E'(w, t) = 2aw(t) + b = 0 \quad (6.2)$$

$$\frac{\partial E(w, t-1)}{\partial w} = E'(w, t-1) = 2aw(t-1) + b = 0 \quad (6.3)$$

$$\frac{\partial E(w, t+1)}{\partial w} = E'(w, t+1) = 2aw(t+1) + b = 0 \quad (6.4)$$

Using Eqn. 6.2 and Eqn. 6.3, it is possible to determine  $2a$ . This is shown in Eqn. 6.5.

$$\begin{aligned} E'(w, t) - E'(w, t-1) &= 2a[w(t) - w(t-1)] \\ 2a &= \frac{E'(w, t) - E'(w, t-1)}{w(t) - w(t-1)} \end{aligned} \quad (6.5)$$

We can also define  $b$  in terms of  $2a$  using Eqn. 6.2.

$$b = E'(w, t) - 2aw(t) \quad (6.6)$$

Finally, substituting the results from Eqn. 6.5 and Eqn. 6.6 into Eqn. 6.4, we can determine the weight update step.

$$\begin{aligned}
w(t+1) &= -\frac{b}{2a} \\
&= -\frac{E'(w,t) - 2aw(t)}{2a} \\
&= w(t) + E'(w,t) \frac{w(t) - w(t-1)}{E'(w,t-1) - E'(w,t)} \\
&= w(t) + \Delta w(t-1) \frac{E'(w,t)}{E'(w,t-1) - E'(w,t)}
\end{aligned} \tag{6.7}$$

When applying this update, we need to ensure that we take extra precaution when the absolute value of the current gradient is larger or equal to the previous gradient, as the update step can diverge to an infinity large number. Therefore, a new parameter  $\mu$  is used to limit the growth rate of the update step. Furthermore, a process is needed to initialise the algorithm and ensure it doesn't cease when a single update step is close to zero. For this, an additional backwards-propagation step is added when the signs of consecutive gradients are equal or when the gradient drops below a certain threshold. This process is explained in Alg. 7.

---

**Algorithm 7** Quick Propagation

---

```

1: procedure QUICKPROPAGATION(input Matrix, Target Matrix,  $\eta$ ,  $\mu$ )
2:   for each  $w$  in the neural networks weights do:
3:      $E'(t) \leftarrow$  The gradient associated with the current weight.
4:      $Scale \leftarrow \frac{E'(t)}{E'(t-1) - E'(t)}$ 
5:      $Scale \leftarrow \text{clip}(Scale, -\mu, \mu)$ 
6:      $\Delta w(t-1) \leftarrow Scale \cdot \Delta w(t-1)$ 
7:     if ( $sign(E'(t-1)) == sign(E'(t))$  or  $E'(t-1) < 1e-15$ ) then
8:        $\Delta w(t-1) \leftarrow \Delta w(t-1) - \eta E'(t)$ 
9:     end if
10:     $w(t) \leftarrow w(t-1) \cdot \Delta w(t-1)$ 
11:   end for
12: end procedure

```

---

## 7 Adaptations to Back-propagation

This section discusses common adaptations to the backward-propagation algorithm.

### 7.1 ADAGRAD

One can improve the backward-propagation algorithm by adjusting the learning rate as the algorithm gets closer to the solution. This is due to the fact that one can take bigger steps when one is further from the optimal solution. The step size can then be decreased to get a better accuracy when one is closer to the optimal solution. One such method is ADAGRAD[10]. This method changes the learning rate on a per weight basis. This is done by keeping track of the sum of squared gradients throughout the optimisation process. The learning rate is then divided by the square-root of this value. Alg. 8 demonstrates this process. Note that the inverse of  $\sqrt{\mathbf{g}_t}$  on line 9 is an element-wise inverse.

---

**Algorithm 8** ADAGRAD

---

```
1: procedure ADAGRAD(input Matrix, Target Matrix,  $\eta$ )
2:   Forward propagate each input and store  $Z^{(m-1)}$  and  $Z^{(m)}$  for each layer  $m$  where  $m \leftarrow 1$  to  $M$ 
3:    $errorContrib \leftarrow \sum_{inputs,targets} (\mathbf{Z}^{(M)} - \mathbf{t})$ 
4:    $\mathbf{g}_\tau \leftarrow \mathbf{0}$ 
5:   for  $m$  from  $M$  down to 1 do
6:     Calculate  $\delta^{(m)}$  using Eqn. 4.14.
7:      $\frac{\partial \mathbf{E}}{\partial \mathbf{W}^{(m)}} \leftarrow$  gradient associated with each weight.
8:      $\mathbf{g}_\tau \leftarrow \mathbf{g}_\tau + \left( \frac{\partial \mathbf{E}}{\partial \mathbf{W}^{(m)}} \right)$ 
9:      $\mathbf{W}^{(m)}(t+1) \leftarrow \mathbf{W}^{(m)}(t) - \frac{\eta}{\sqrt{\mathbf{g}_\tau}} \delta^{(m)} \cdot (\mathbf{Z}^{(m-1)})^T$ 
10:     $errorContrib \leftarrow (\mathbf{W}^{(m)} \cdot \delta^{(m)})$ 
11:   end for
12: end procedure
```

---

## 7.2 Momentum

Backward-propagation is essentially gradient descent on the weights of a neural network. Due to this, it is not guaranteed to find the global optimal solution. One method, which attempts to reduce the chances of the algorithm settling on a local minimum, is to add momentum to the update process. This helps backward propagation as it allows it to deteriorate in accuracy slightly in order to potentially locate a better solution. A momentum term also helps speed up convergence as it effectively increases the learning rate when there are multiple updates in the same direction[11].

---

**Algorithm 9** Backward-Propagation with Momentum

---

```
1: procedure BACKWARD-PROPAGATION WITH MOMENTUM(input Matrix, Target Matrix,
 $\eta, \alpha$ )
2:   Forward propagate each input and store  $Z^{(m-1)}$  and  $Z^{(m)}$  for each layer  $m$  where  $m \leftarrow 1$  to  $M$ 
3:    $errorContrib \leftarrow \sum_{inputs,targets} (\mathbf{Z}^{(M)} - \mathbf{t})$ 
4:    $\Delta \mathbf{W}^{(n)} \leftarrow \mathbf{0}$ 
5:   for  $m$  from  $M$  down to 1 do
6:     Calculate  $\delta^{(m)}$  using Eqn. 4.14.
7:      $\frac{\partial \mathbf{E}}{\partial \mathbf{W}^{(m)}} \leftarrow$  gradient associated with each weight.
8:      $\Delta \mathbf{W}^{(n)} \leftarrow -\eta \delta^{(m)} \cdot (\mathbf{Z}^{(m-1)})^T + \alpha \Delta \mathbf{W}^{(n)}$ 
9:      $\mathbf{W}^{(m)}(t+1) \leftarrow \mathbf{W}^{(m)}(t) + \Delta \mathbf{W}^{(n)}$ 
10:     $errorContrib \leftarrow (\mathbf{W}^{(m)} \cdot \delta^{(m)})$ 
11:   end for
12: end procedure
```

---

## 8 Genetic Algorithm

The Genetic Algorithm (GA) is an optimisation algorithm that does not depend on a derivative of the error function. Due to this, it is able to optimise complex problems, where one is unable to obtain gradient information. This algorithm attempts to mimic stages in evolution as described by Darwin[12]. GA starts with an initial population of creatures each representing an initial configuration of parameters. These parameters are usually represented as a string of binary digits to represent chromosomes in the "genes" of each creature. However, it is also possible to separate each parameter as a chromosome, making the gene an array of numbers as opposed to a binary string. Each creature is then evaluated and a selection process is used to pick a top proportion of the population. These creatures are then used to generate the next generation of creatures in a process referred to as crossover. Finally, a small portion of the chromosomes are changed in order to add diversity into the population. This process is referred to as mutation.

### 8.1 Selection

The most simplistic method of selection chooses the top  $M$  creatures based on their errors. This process is also very efficient as the individual creatures within the top  $M$  do not need to be in any particular order. An alternative method is to select each creature based on a probability proportional to their inverse error. A small twiddle factor is added to this error to allow numeric stability. This selection process is more computationally expensive but allows for slightly worse creatures to be selected. Therefore, the probabilistic approach is more likely to converge to more optimal solutions than the simplistic method as it is less likely to get stuck on local minima.

### 8.2 Crossover

There is a vast variety of crossover algorithms. The simplest of which, breaks each gene of two parents in half and then recombines them to form two new genes. We can then have every creature crossover with every other creature, creating a population which contains all the combinations of the previously selected population. However, this lessens the diversity of the population as a small set of the overall population must be selected,  $M = \sqrt{N}$ , in order to maintain a constant population size. Furthermore, a single gene in the parent has influence over a large population of the child genes.

A second approach is to combine two parents which are selected consecutively. This can be done in a circular array fashion to produce a child population which is twice the size of the selected population. This approach allows a large portion of the initial population to be selected, increasing the diversity of the overall population. This is important to allow the algorithm to converge to an optimal solution.

### 8.3 Mutation

Mutation is used to add diversity onto the population so that new parameters can be searched. Assuming that the population is represented as an array of floats, one can achieve this by selecting  $p$  chromosomes from the entire population. Gaussian noise is then added to these chromosomes with the form of  $\mathcal{N}(0, \sigma^2)$ . Throughout the optimisation process,  $\sigma^2$  can be reduced to allow the algorithm to be more random at the start and more selective when closer to an optimal solution. This can be done simply by defining a minimal variance  $\sigma_{min}^2$  and then multiplying the variance by a fixed factor  $0 < c < 1$  each iteration, ensuring that  $\sigma^2 \geq \sigma_{min}^2$ .

## 9 Particle Swarm Optimisation

Particle Swarm Optimisation (PSO)[13, 14] attempts to mimic swarm intelligence in order to improve the convergence rate to an optimal solution. The PSO algorithm consists of a set of particles with an initial velocity. Each iteration the particles evaluate the performance of their current position. The global best position, and per-particle best positions are stored. The particles velocity is then updated in order to direct it to either the local or global best position. This velocity update step is controlled by 2 parameters  $\Phi_p$  and  $\Phi_g$ , which control how much the particle should be directed to the local and global best positions respectively. The uniformly distributed random numbers  $r_g$  and  $r_p$  are also used to add some randomness to this process. Pseudopod for PSO is shown in Alg. 10.

---

**Algorithm 10** PSO

---

```
1: procedure PSO( $\Phi_g$ ,  $\Phi_p$ ,  $\omega$ , Initial particle positions, Position Range)
2:   particlePositions  $\leftarrow$  Initial particle positions
3:   bestPositions  $\leftarrow$  Initial particle positions
4:   bestValues  $\leftarrow$  Error score for each particle at current position.
5:   globalBestIndex  $\leftarrow$  index of the lowest value in bestValues.
6:    $\mathbf{V} \leftarrow \mathbf{U}(-\text{Position Range}, \text{Position Range})$ 
7:   while not converged do
8:     for each particle p do
9:        $\mathbf{r}_p \leftarrow \mathbf{U}(0, 1)$ 
10:       $\mathbf{r}_g \leftarrow \mathbf{U}(0, 1)$ 
11:       $\mathbf{V} \leftarrow \omega \mathbf{V} + \Phi_p \mathbf{r}_p \odot (\text{bestPositions}[index_p] - \text{particlePositions}[index_p]) + \Phi_g \mathbf{r}_g \odot$ 
         $(\text{bestPositions}[globalBestIndex] - \text{particlePositions}[index_p])$ 
12:       $\text{particlePositions}[index_p] \leftarrow \text{particlePositions}[index_p] + \mathbf{V}$ 
13:      error  $\leftarrow$  error of particle at current position.
14:      if error < bestValues[indexp] then
15:        bestValues[indexp]  $\leftarrow$  error
16:        bestPositions[indexp]  $\leftarrow$  particlePositions[indexp]
17:        if error < bestValues[globalBestIndex] then
18:          globalBestIndex  $\leftarrow$  indexp
19:        end if
20:      end if
21:    end for
22:  end while
23: end procedure
```

---

## 10 Proben1

Traditionally neural network optimisers have been benchmarked on simplistic problems such as mimicking a logical XOR gate. This provides useful information on convergence rates but does not provide an indication on how the algorithm copes with outliers and missing data. The Proben1 dataset[15] provides a set of real world problems which contain outliers and missing information. This dataset is extensively studied and thus provides a useful platform, which can be used to both test and benchmark various algorithms. This dataset is fairly extensive, containing 15 different problems. Therefore it was decided to use a subset of these problems in order to evaluate the algorithms discussed in this paper. These are summarised in Tbl. 1, where the number of inputs, outputs and items are displayed. The multi-layered networks which produced the best results in [15] are also shown.

<b>Dataset Name</b>	<b>Number of Inputs</b>	<b>Number of Outputs</b>	<b>Number or items</b>	<b>Tested Network Architecture</b>	<b>Description</b>
<b>Cancer</b>	9	2	699	$4 \times 2$	Determine if a cancer is benign or malignant
<b>Card</b>	51	2	690	$4 \times 4$	Determine if a credit card should be granted or denied
<b>Flare</b>	24	3	1066	4	Determine the number of small, medium and large solar flares that will occur in the next 24 hours
<b>Gene</b>	120	3	3175	$4 \times 2$	Determine if a DNA sequence is a donor, a acceptor or neither.
<b>Heartc</b>	35	2	303	$8 \times 8$	Determine if at least one of four major vessels of the heart is reduced in diameter by more than 50%.
<b>Horse</b>	58	3	364	4	Determine whether a horse, which has colic, will survive, die or be euthanized.

**Table 1.** Summary of datasets tested

## 11 Evaluation Procedure

This section discusses how the algorithms are to be evaluated. For all these tests, the dataset is split into a training, validation and testing dataset. The training dataset is then split into mini-batches, which are then used in order to train the network. This report refers to an Epoch as an entire iteration through all the mini-batches belonging to a training set (equivalently the number of times a given input has been through the training algorithm). After each mini-batch, the validation error, training error and optionally the testing error is evaluated using the applicable datasets. This is done with the use of Eqn. 11.1, as done in [15]. This is known as the squared error percentage and allows the error to be more comparable over various datasets.

$$E = 100 \cdot \frac{o_{max} - o_{min}}{N \cdot P} \sum_{p=1}^P \sum_{i=1}^N (o_{pi} - t_{pi})^2 \quad (11.1)$$

In Eqn. 11.1, we define  $o_{max}$  and  $o_{min}$  as the maximum and minimum outputs of a node respectively. The parameters  $N$  and  $P$  refer to the number of outputs and the number of examples respectively.

It is also important to know when the algorithm is finished training. For this, we use the  $PQ_\alpha$  stopping criteria[16]. For this we define a generalization loss as shown in Eqn. 11.2. This gives a measure of how much the algorithm has lost its ability to generalise well to unseen data. A parameter to track the training progress is also defined as shown in Eqn. 11.3. This parameter stops the algorithm terminating when the training error is still declining rapidly, as this may cause premature termination. We define a window of  $k = 5$  epochs, which are used to calculate the training progress. The algorithm is then

terminated when  $GL(t) > \alpha P_k(t)$ . Unless otherwise specified, we define  $\alpha$  to be 0.5.

$$GL(t) = 100 \cdot \left( \frac{E_{va}(t)}{E_{opt}(t)} - 1 \right) \quad (11.2)$$

$$P_k(t) = 1000 \cdot \left( \frac{\sum_{t'=t-k+1}^t E_{tr}(t')}{k \cdot \min_{t'=t-k+1}^t E_{tr}(t')} - 1 \right) \quad (11.3)$$

The various parameters of the algorithms used are summarised in Tbl. 2.

Algorithm	Parameters
<b>Backward Propagation</b>	$\eta$ exponentially decays from 0.8 at epoch 1 to 0.005 at epoch 25 and higher
<b>RPROP</b>	$\eta_- = 0.5$ $\eta_+ = 1.2$ $\Delta_{min} = 0$ $\Delta_{max} = 50$ $\Delta = \mathcal{U}(0.005, 0.2)$
<b>QuickProp</b>	$\eta = 2$ $\mu = 1.75$
<b>ADAGRAD</b>	$\eta = 3$
<b>Back Propagation with Momentum</b>	$\eta$ exponentially decays from 0.8 at epoch 1 to 0.005 at epoch 25 and higher $\alpha = 0.2$
<b>GA1</b>	Number of creatures = 64 Selection: Simplistic Crossover: All Combinations Mutation: Select $p = 5\%$ of chromosomes Gaussian with $\sigma_{initial}^2 = 1$ , $\sigma_{min}^2 = 0.05$ and $c = 0.9$
<b>GA2</b>	Number of creatures = 64 Selection: Probabilistic Crossover: All Combinations Mutation: Select $p = 5\%$ of chromosomes Gaussian with $\sigma_{initial}^2 = 1$ , $\sigma_{min}^2 = 0.05$ and $c = 0.9$
<b>GA3</b>	Number of creatures = 64 Selection: Simplistic Crossover: Nearest 2 Mutation: Select $p = 5\%$ of chromosomes Gaussian with $\sigma_{initial}^2 = 1$ , $\sigma_{min}^2 = 0.05$ and $c = 0.9$
<b>PSO</b>	Number of particles = 64 $Phi_g = 1.49618$ $Phi_p = 1.49618$ $\omega = 0.7298$ These parameters were chosen from [17]

**Table 2.** Summary of experimental parameters

## 11.1 Experiment 1: Testing with Proben1

### 11.1.1 Aim

To evaluate the above algorithms on the Proben1 dataset. Furthermore, the rate of convergence, accuracy and resiliency of each algorithm is to be compared.

### 11.1.2 Procedure

The following is performed in order to obtain the results:

1. A selected dataset is shuffled and then split up into a test set, a training set and a validation set using the proportions provided in the header section of the dataset.
2. The dataset is then split up into mini-batches containing no more than 30 items.
3. A selected algorithm and layer configuration is used to train the neural network on the mini-batch.
4. The test error, validation error and training error are calculated after each mini-batch.
5. The validation error s and training errors are use to determine if the algorithm is to be terminated using the  $PQ_{0.5}$  stopping conditions.
6. If the termination criteria is met, the epoch count is recorded. Furthermore, if this is the first trial, the maximum epoch count is set to 1.6 times the current epoch count.
7. The algorithm is let to run to its maximum epoch count to allow all trials to have the same number of data-points. It should be noted that the termination epoch count is no longer allowed to update for this trial.
8. The best validation error, test error at the best validation error, epochs to best validation error and time taken to achieve the best validation error is recorded.
9. This process is repeated for 30 trials after which the average and standard deviation of the test errors, training errors and validation error vs epoch are plotted until the upper quartile on the number of epochs until termination. This allows all the graphs to only show the relevant data. The median, upper quartile and lower quartile is also plotted, in order to provide better insight to how the training is skewed around the median. finally the box and whisker diagrams for the data collected in step 8 is recorded.
10. This is then repeated for each dataset tested, network configuration and training algorithm.

## 11.2 Experiment 2: MNIST

### 11.2.1 Aim

To evaluate the effectiveness of a neural network on the MNIST dataset and compare its performance to a naive Bayesian classifier.

### 11.2.2 Procedure

The following is performed in order to obtain the results:

1. The training dataset, provided by MNIST[1], is randomly split up into a training set (containing 90% of the images) and a validation set (containing 10% of the images).
2. The training set is split into mini-batches containing 200 images.
3. A small amount of Gaussian noise is added to the original training images each epoch in order to make the final output more resilient to change.
4. iRPORP is used to train the neural network ( $784 \times 128 \times 10$ ).
5. The  $PQ_{0.5}$  stopping criteria is used to determine when to stop training.
6. The weights which produced the best results on the validation dataset is then used to test the classifier on the testing dataset provided by MNIST.

## 12 Results

This section summarises the results of the experiments performed.

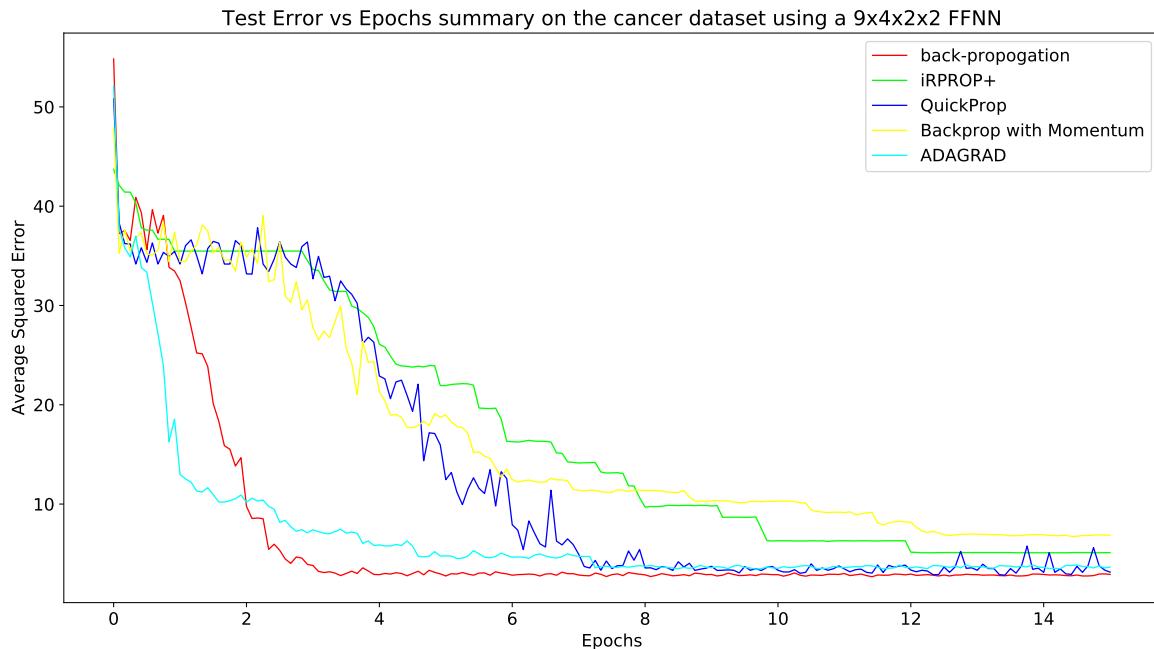
### 12.1 Experiment 1: Testing with Proben1

This section shows only a summary of the most relevant data. For more detailed results please refer to the appendix.

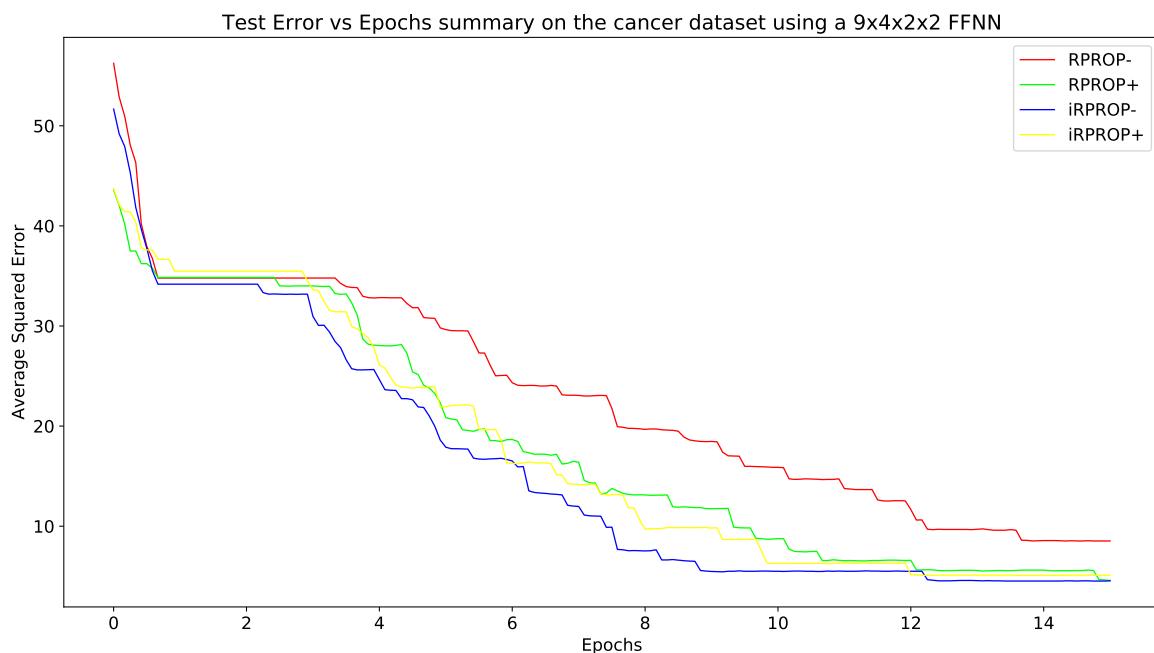
### 12.1.1 Cancer dataset

This section summarises the results obtained on the cancer dataset.

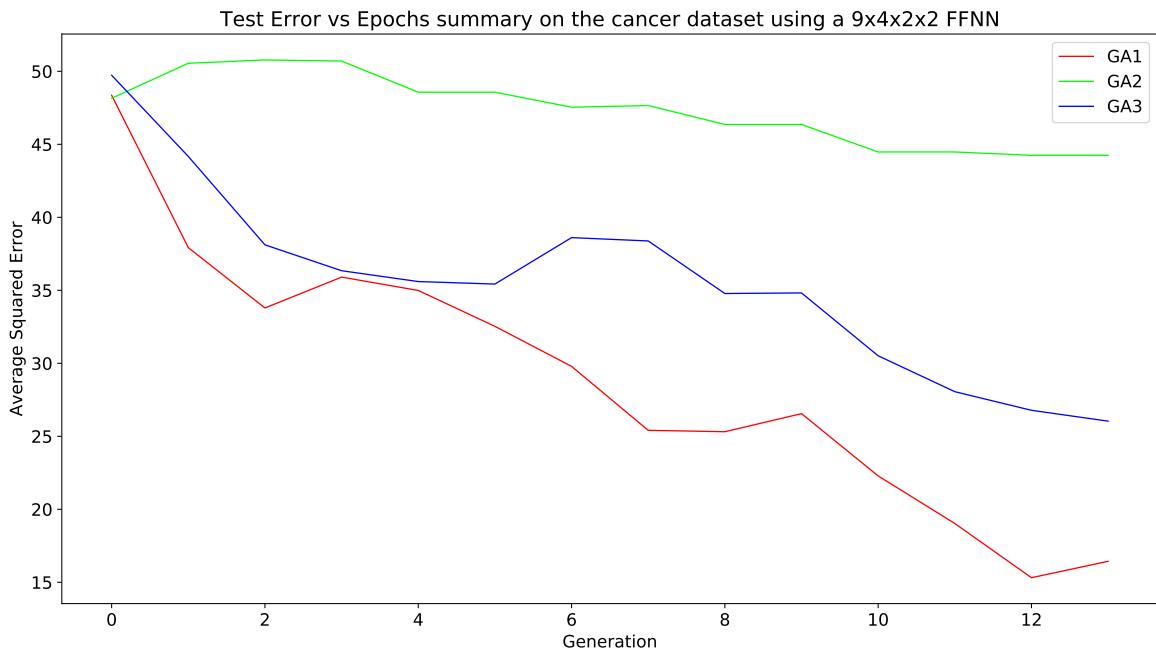
#### **$9 \times 4 \times 2 \times 2$ Architecture:**



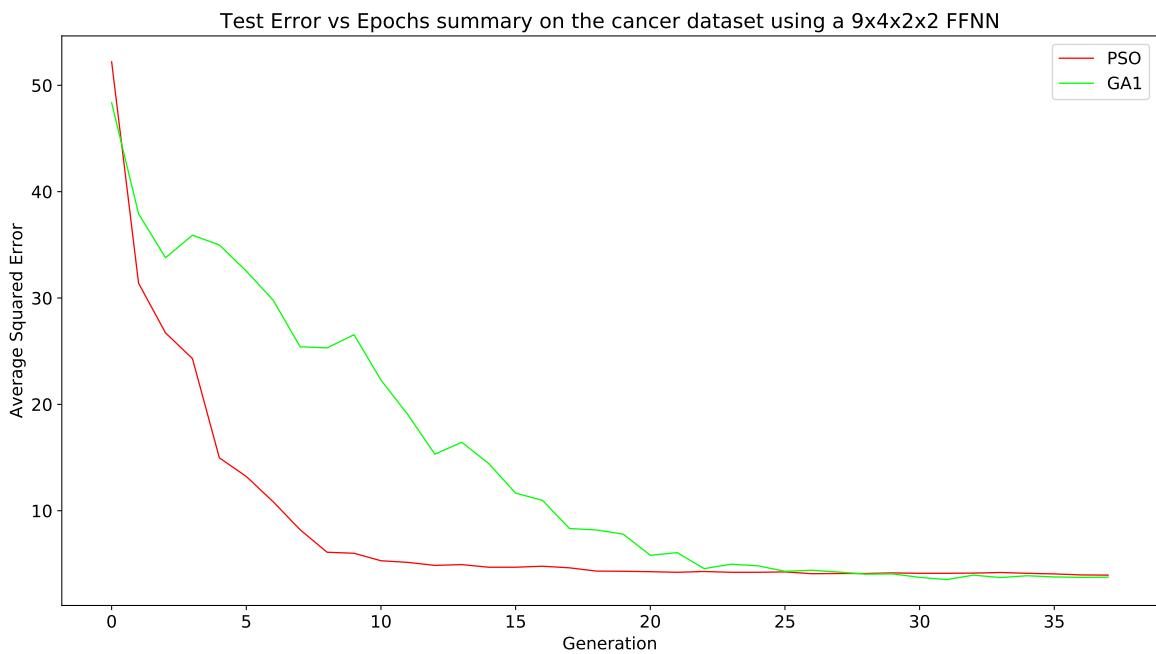
**Figure 4.** Graph of test error vs epochs for the gradient based algorithms



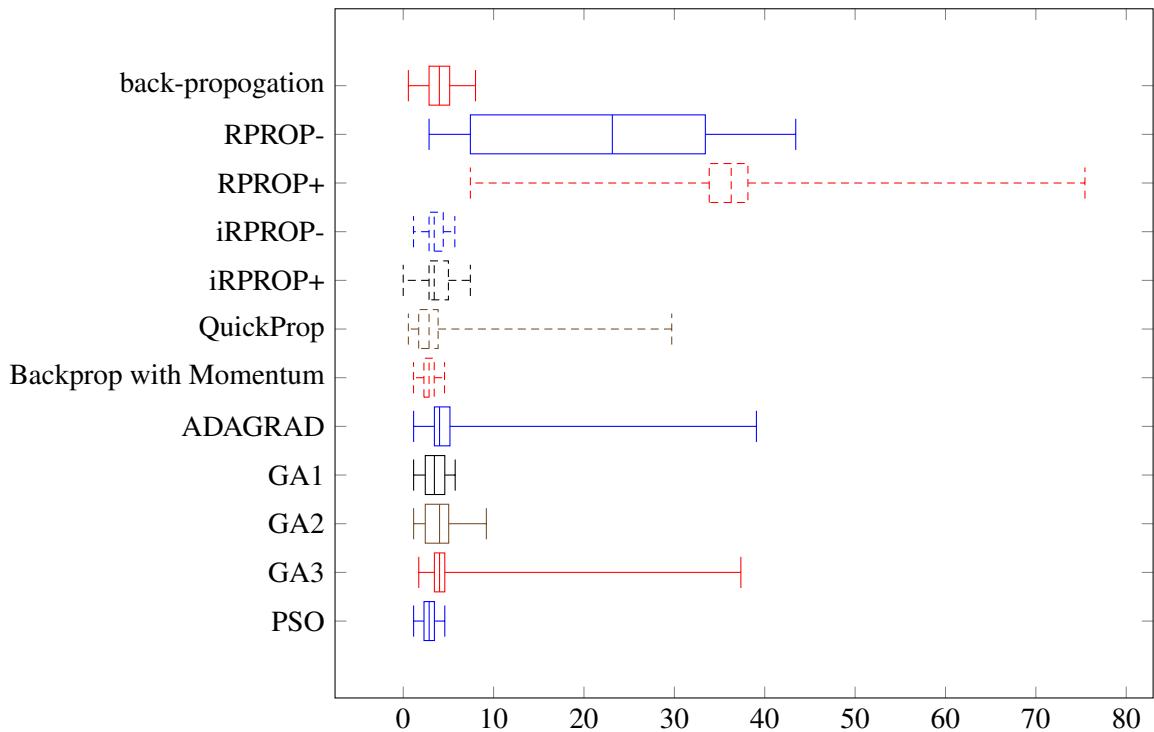
**Figure 5.** Graph of test error vs epochs for the various version of RPROP



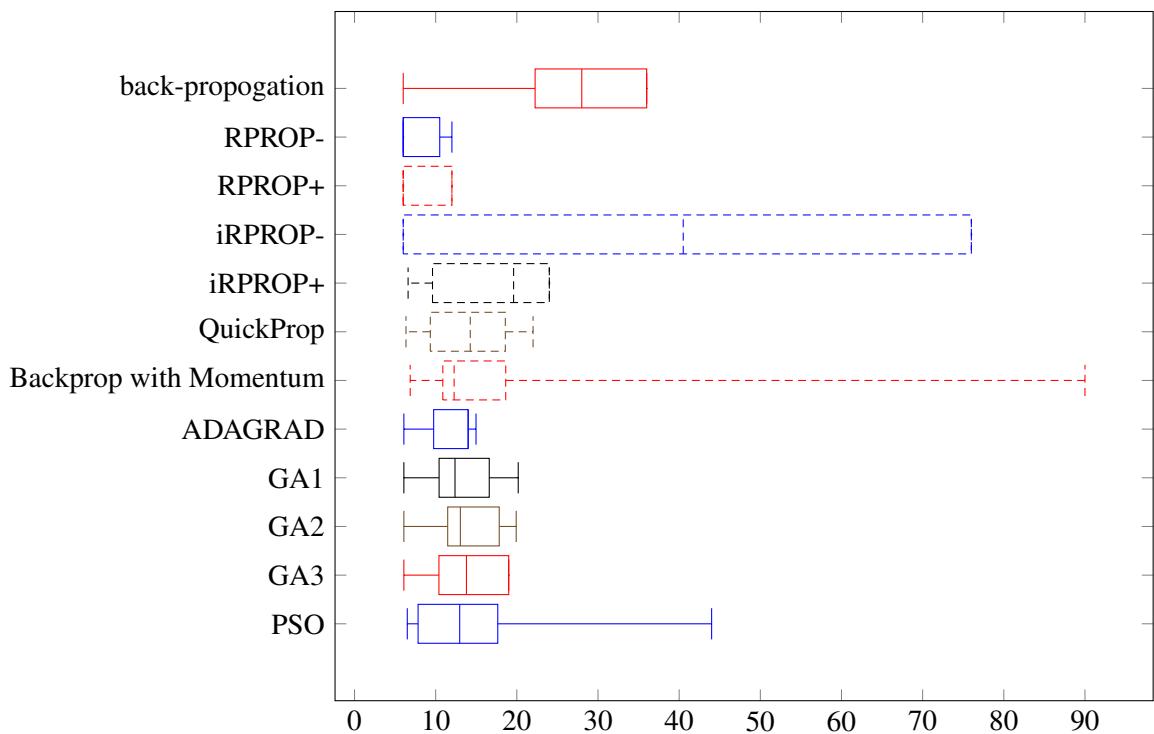
**Figure 6.** Graph of test error vs epochs for the genetic algorithms



**Figure 7.** Graph comparing GA and PSO

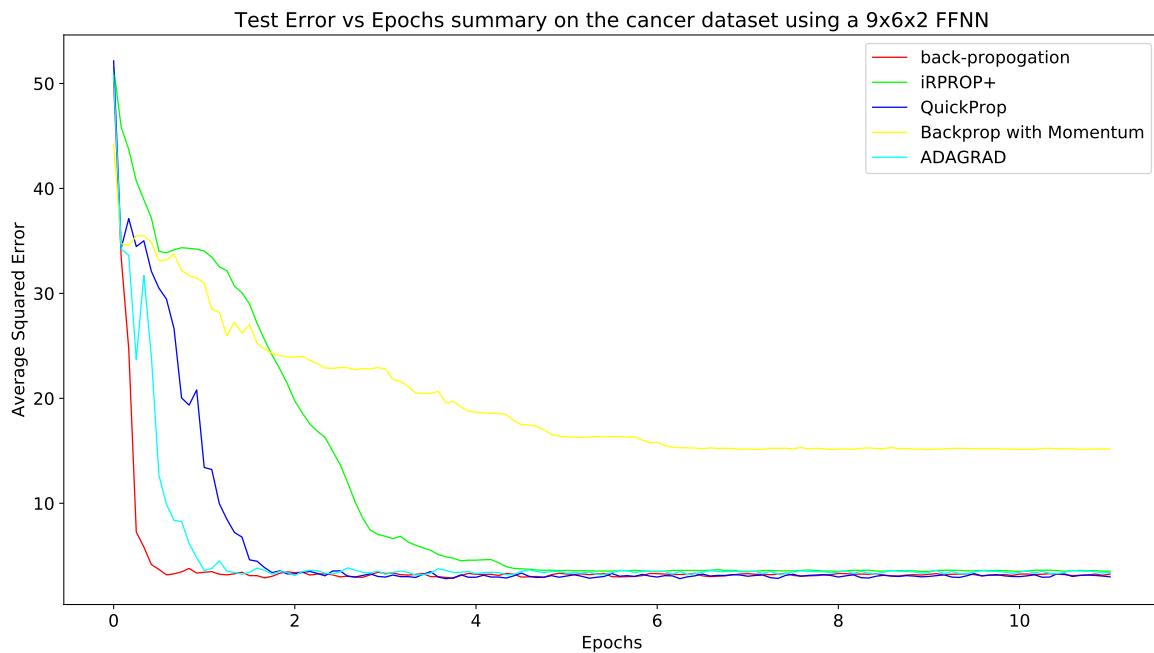


**Figure 8.** Best test errors achieved

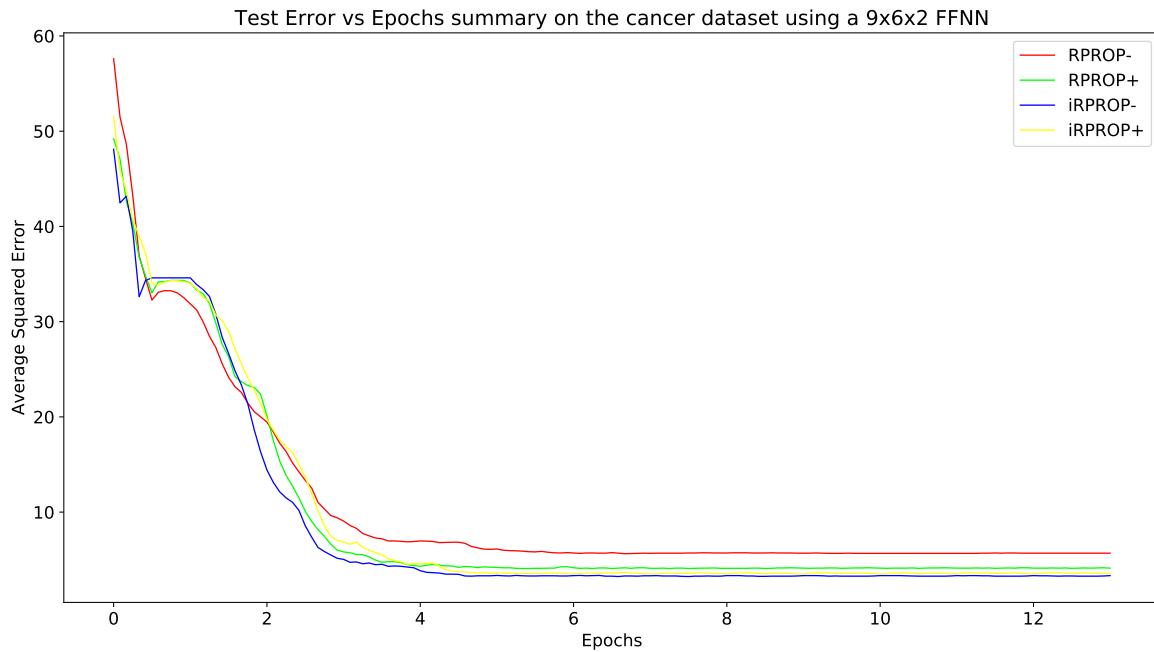


**Figure 9.** Number of epochs until algorithm terminated

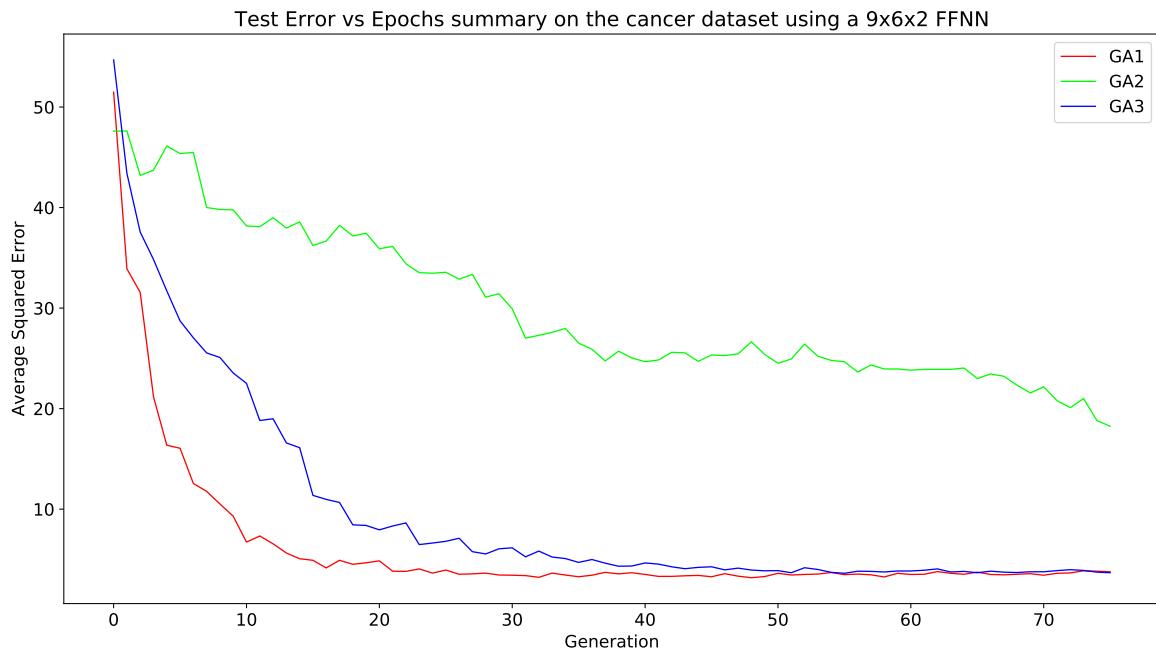
### **$9 \times 6 \times 2$ Architecture:**



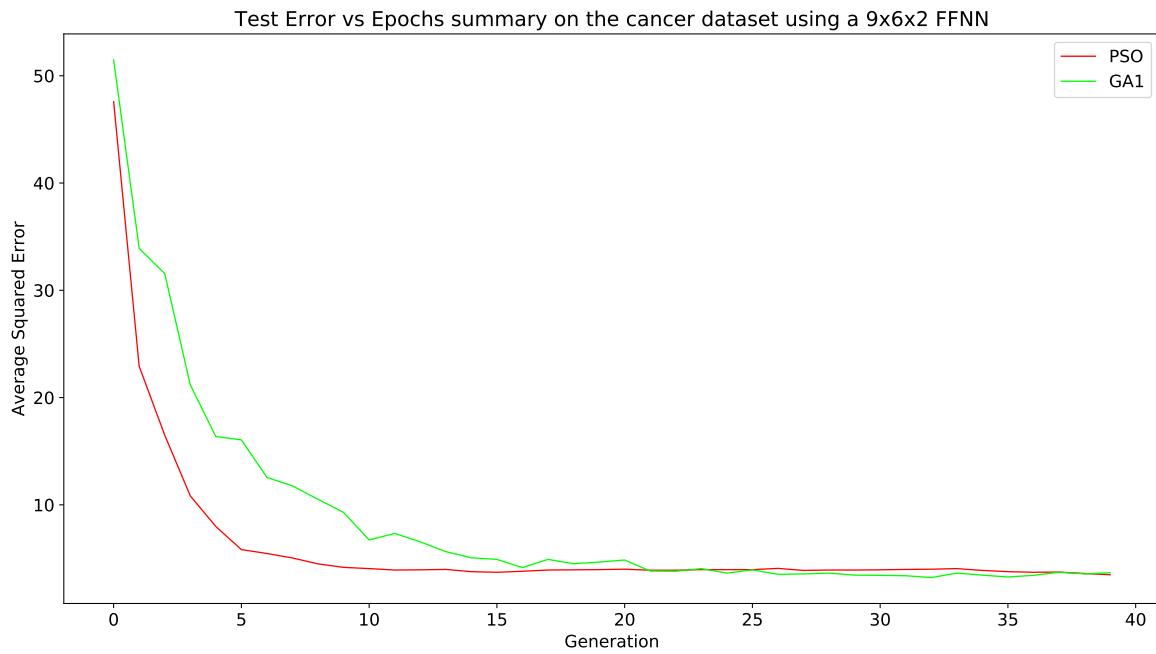
**Figure 10.** Graph of test error vs epochs for the gradient based algorithms



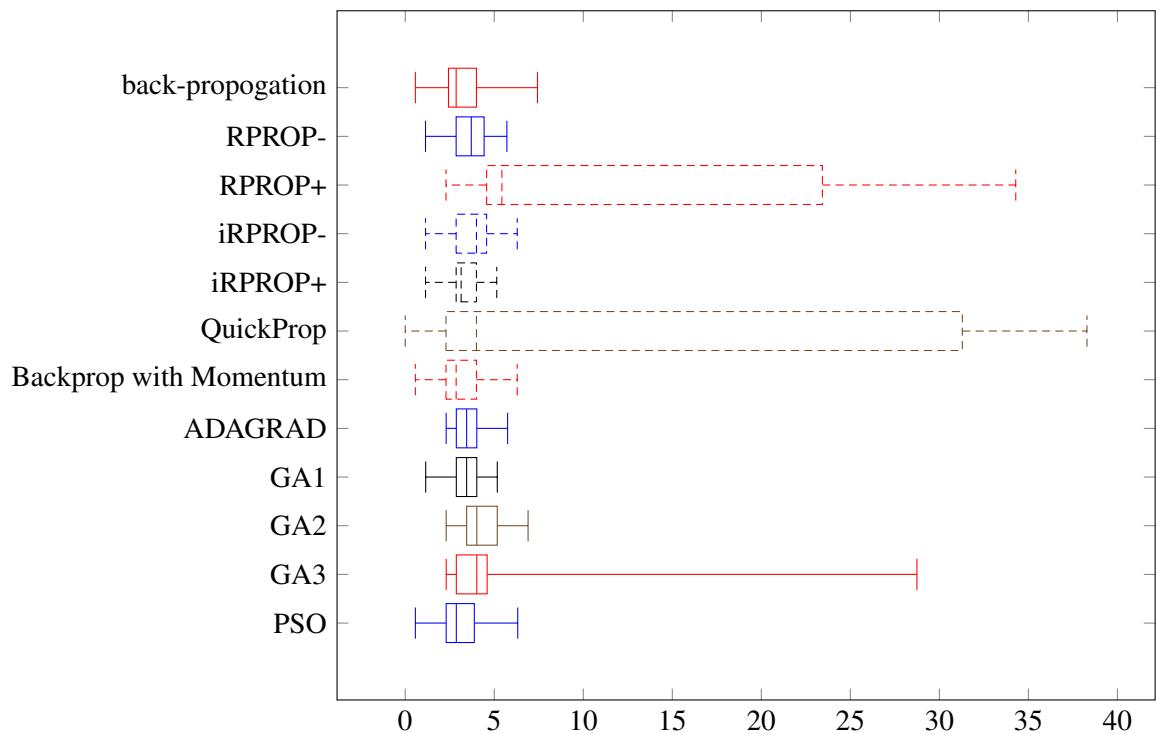
**Figure 11.** Graph of test error vs epochs for the various version of RPROP



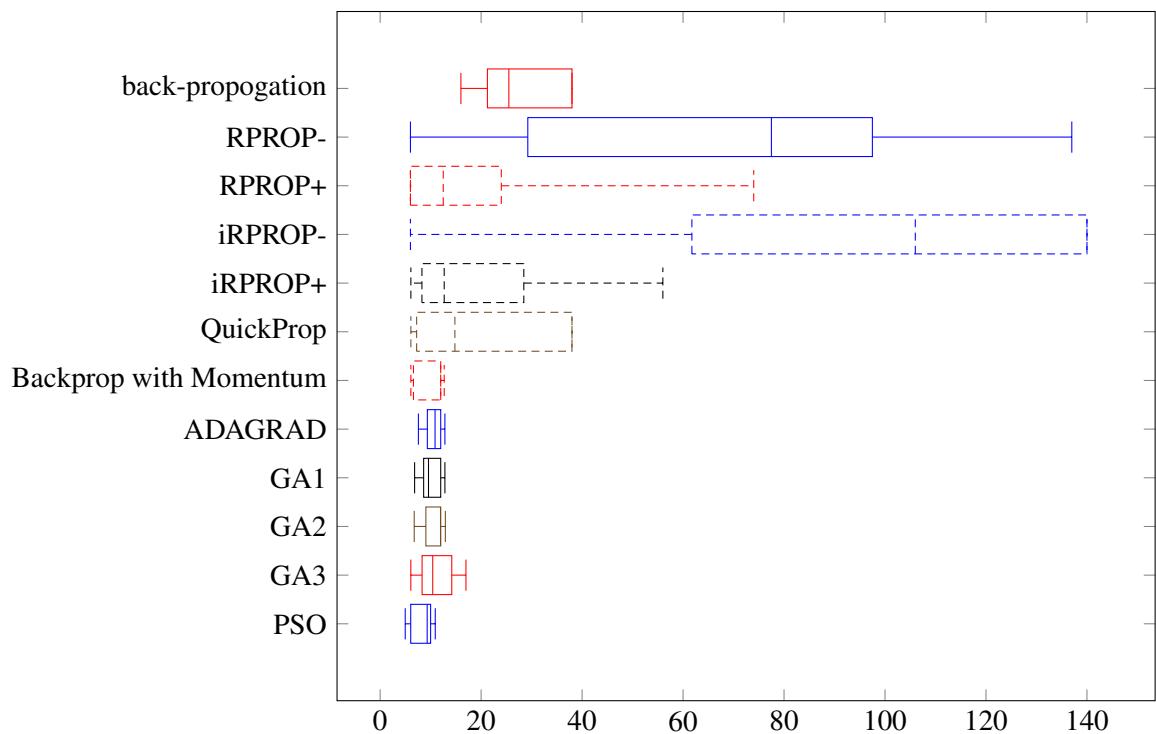
**Figure 12.** Graph of test error vs epochs for the genetic algorithms



**Figure 13.** Graph comparing GA and PSO



**Figure 14.** Best test errors achieved

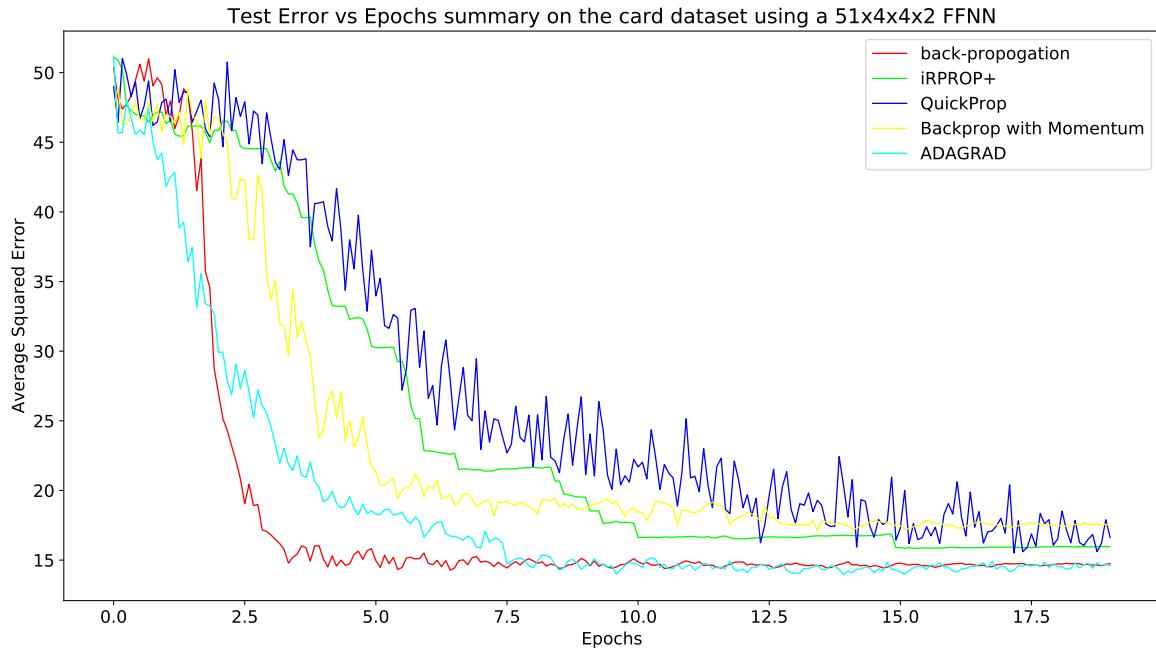


**Figure 15.** Number of epochs until algorithm terminated

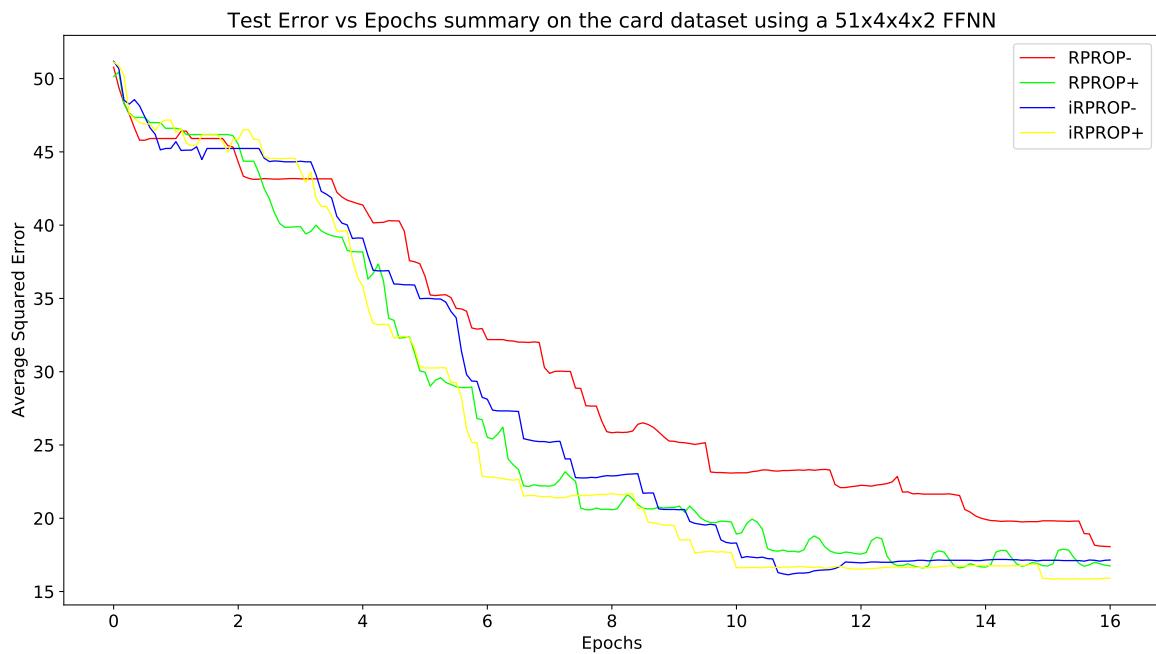
### 12.1.2 Card dataset

This section summarises the results obtained on the card dataset.

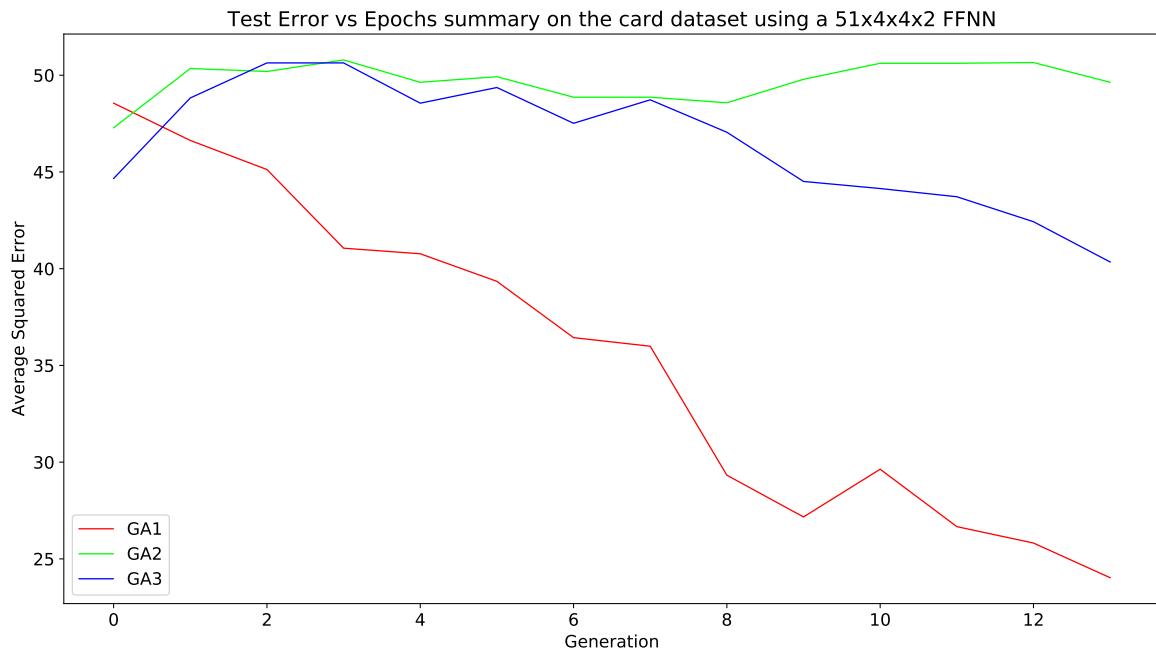
#### **51 × 4 × 4 × 2 Architecture:**



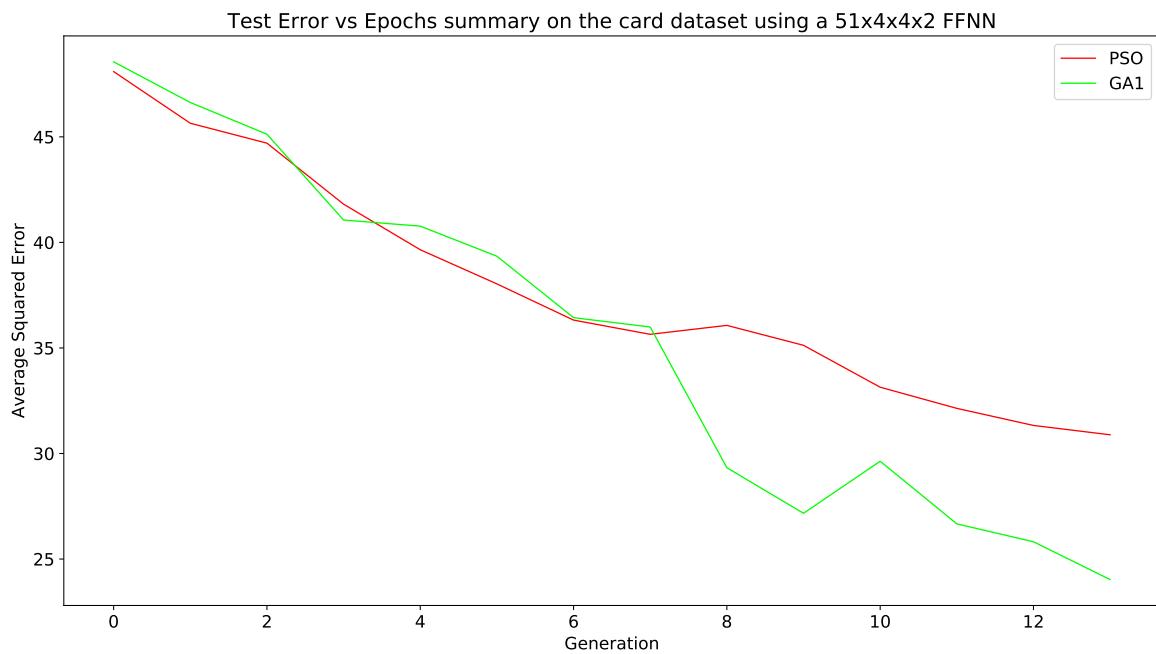
**Figure 16.** Graph of test error vs epochs for the gradient based algorithms



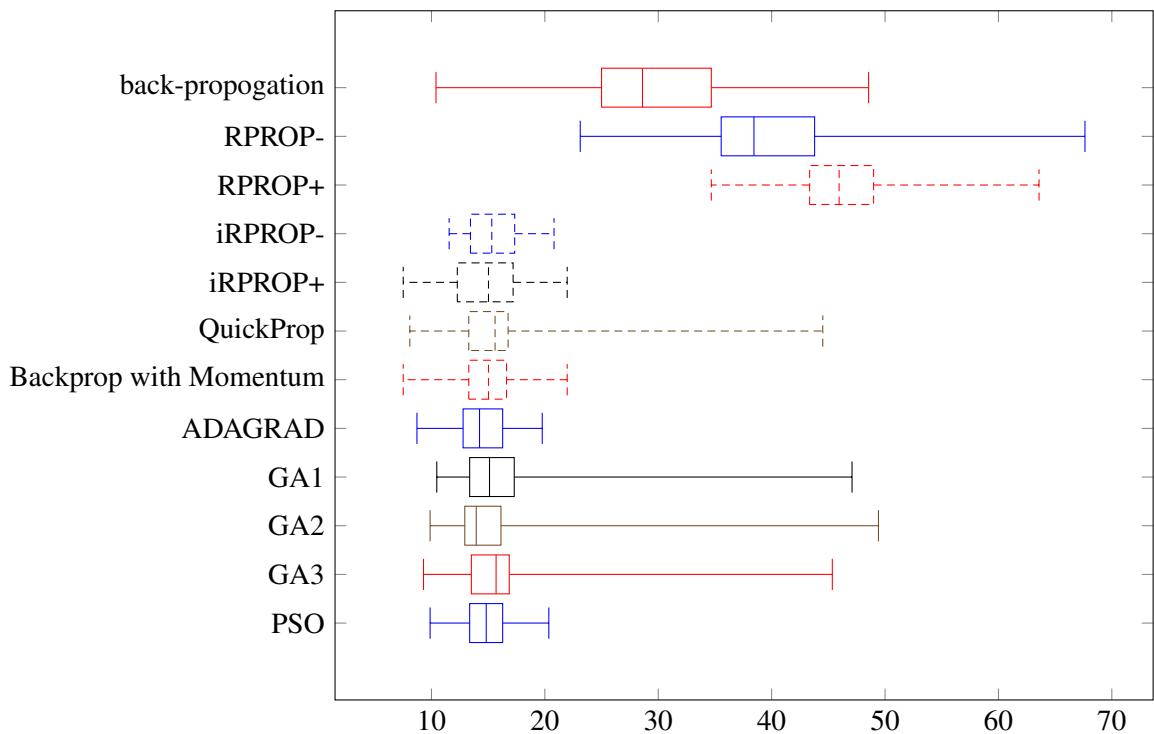
**Figure 17.** Graph of test error vs epochs for the various version of RPROP



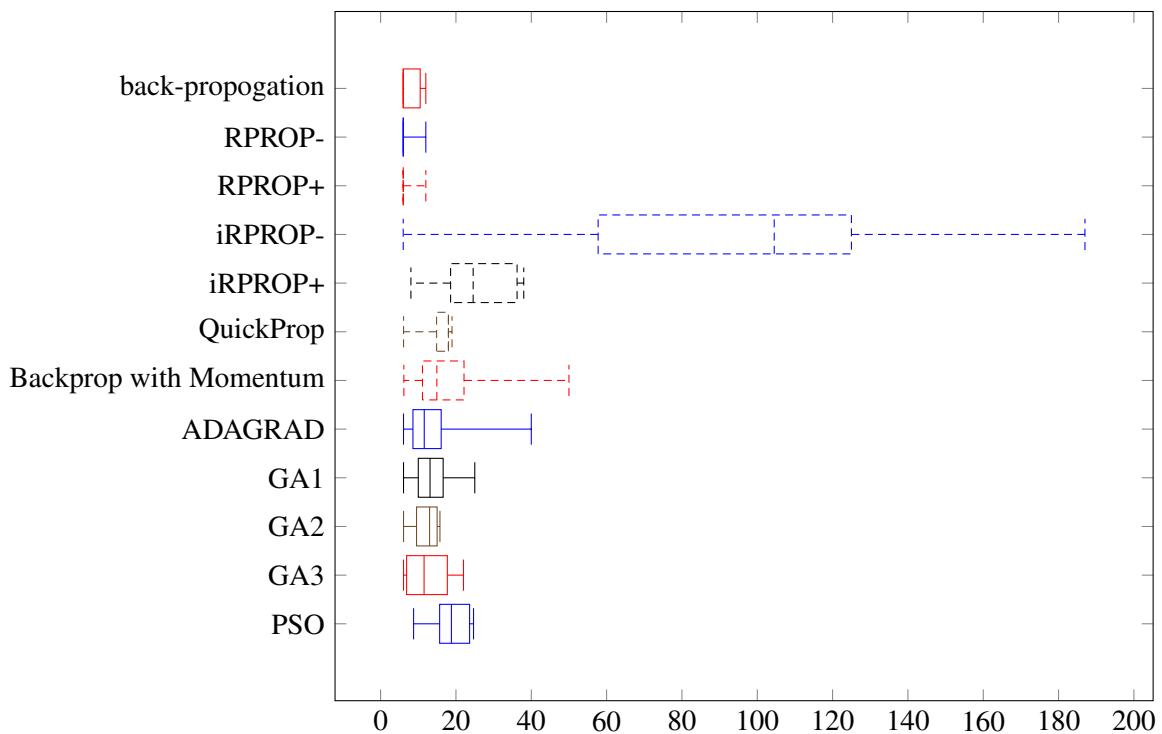
**Figure 18.** Graph of test error vs epochs for the genetic algorithms



**Figure 19.** Graph comparing GA and PSO

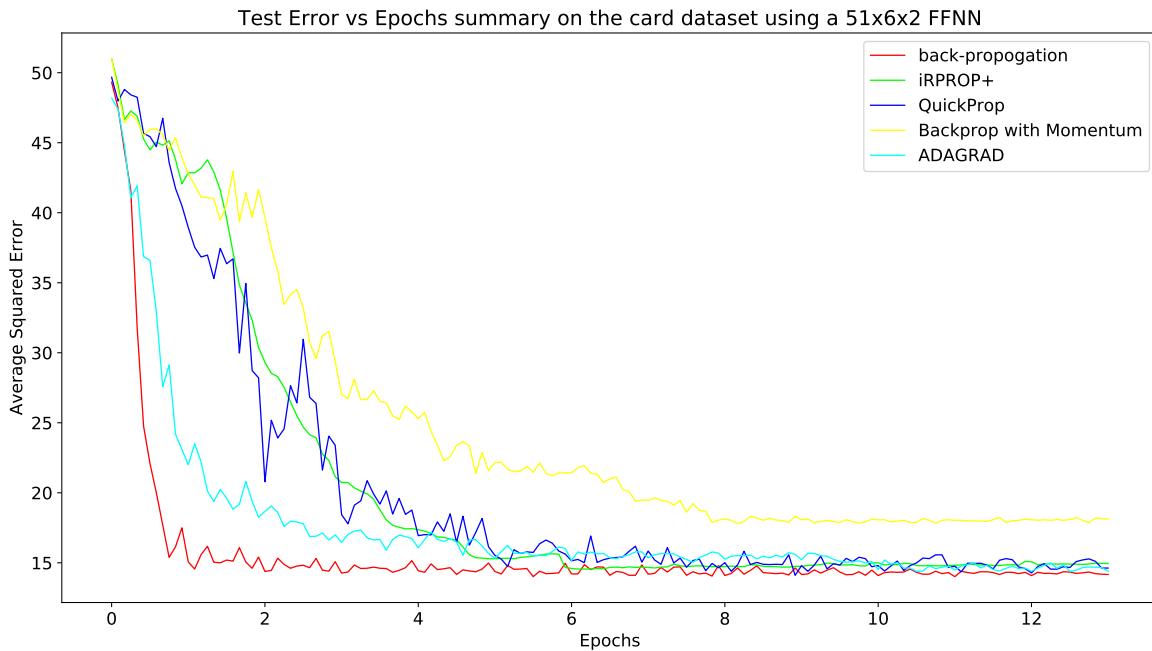


**Figure 20.** Best test errors achieved

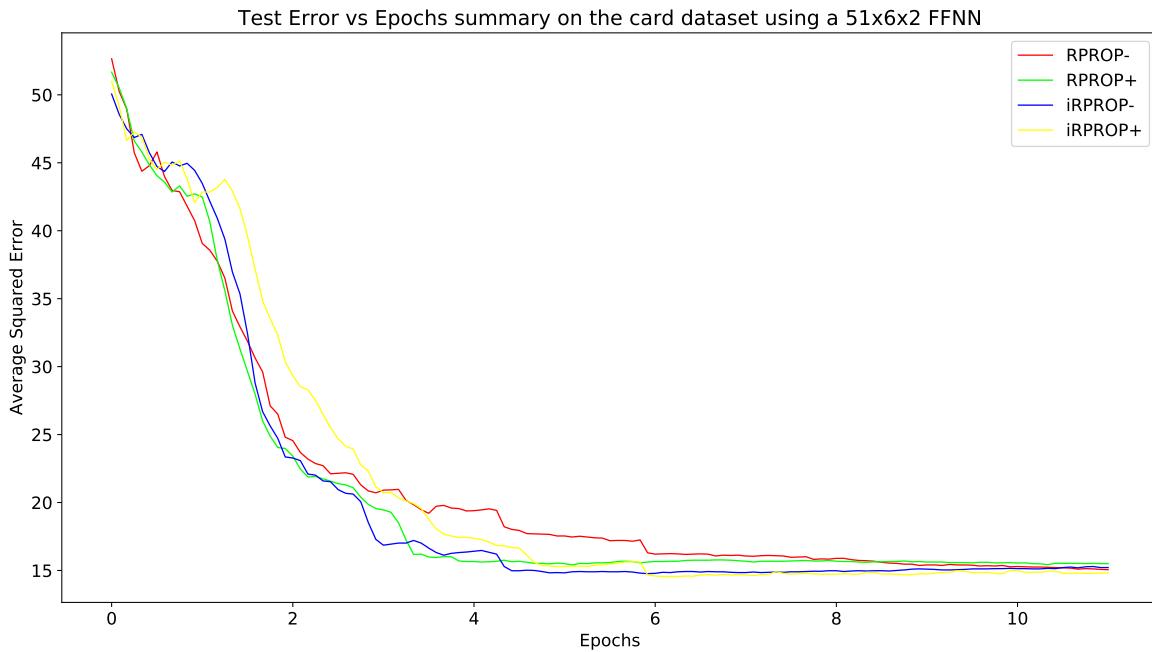


**Figure 21.** Number of epochs until algorithm terminated

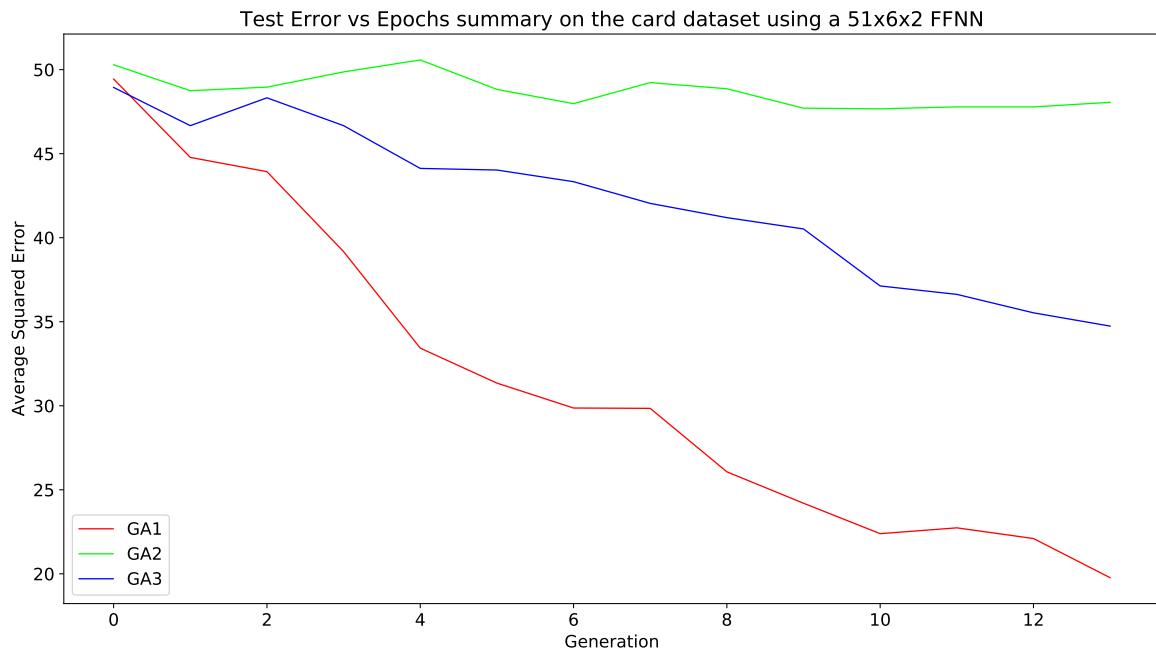
### **51 × 6 × 2 Architecture:**



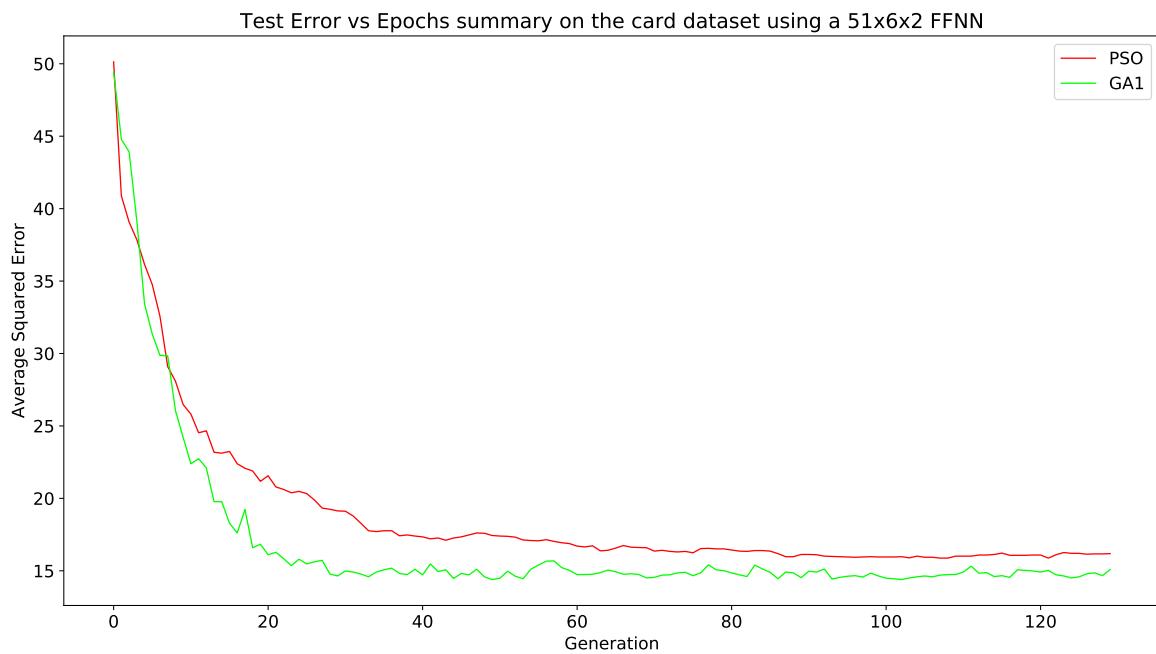
**Figure 22.** Graph of test error vs epochs for the gradient based algorithms



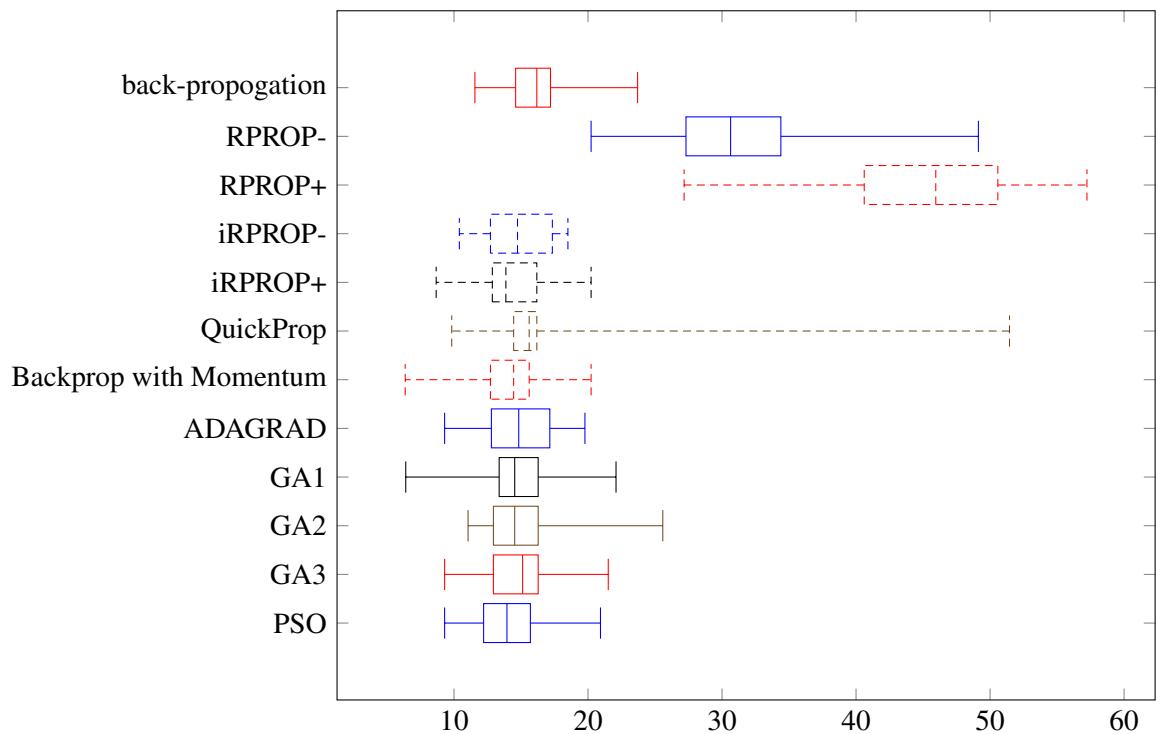
**Figure 23.** Graph of test error vs epochs for the various version of RPROP



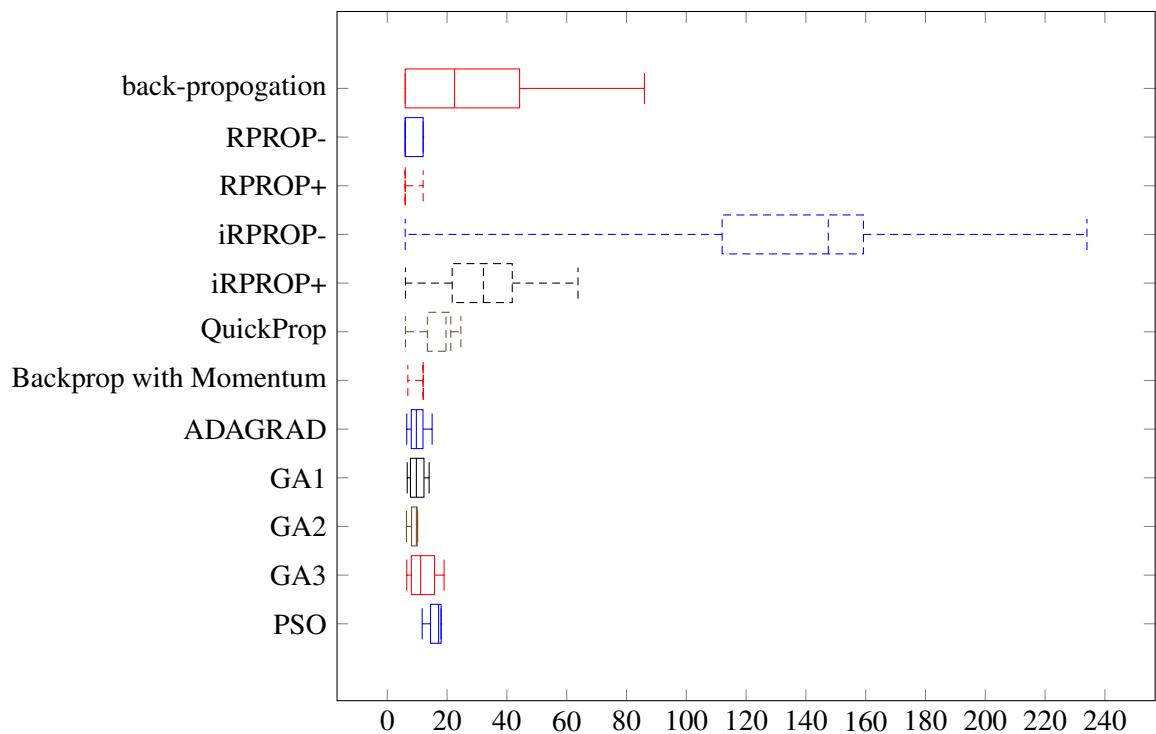
**Figure 24.** Graph of test error vs epochs for the genetic algorithms



**Figure 25.** Graph comparing GA and PSO



**Figure 26.** Best test errors achieved

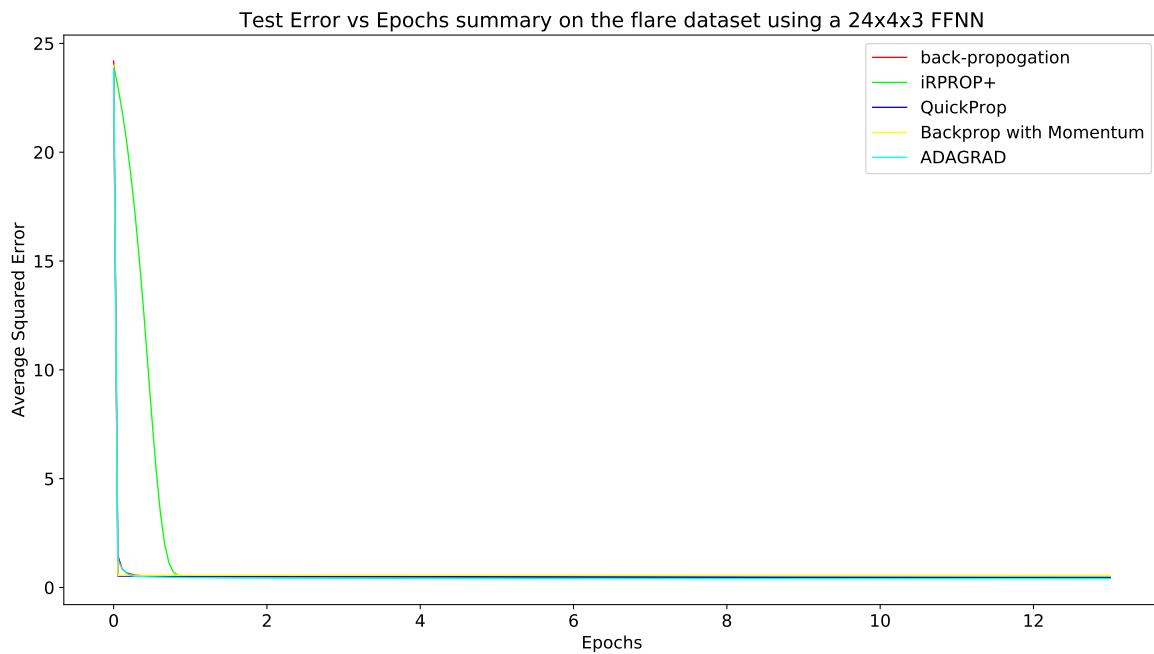


**Figure 27.** Number of epochs until algorithm terminated

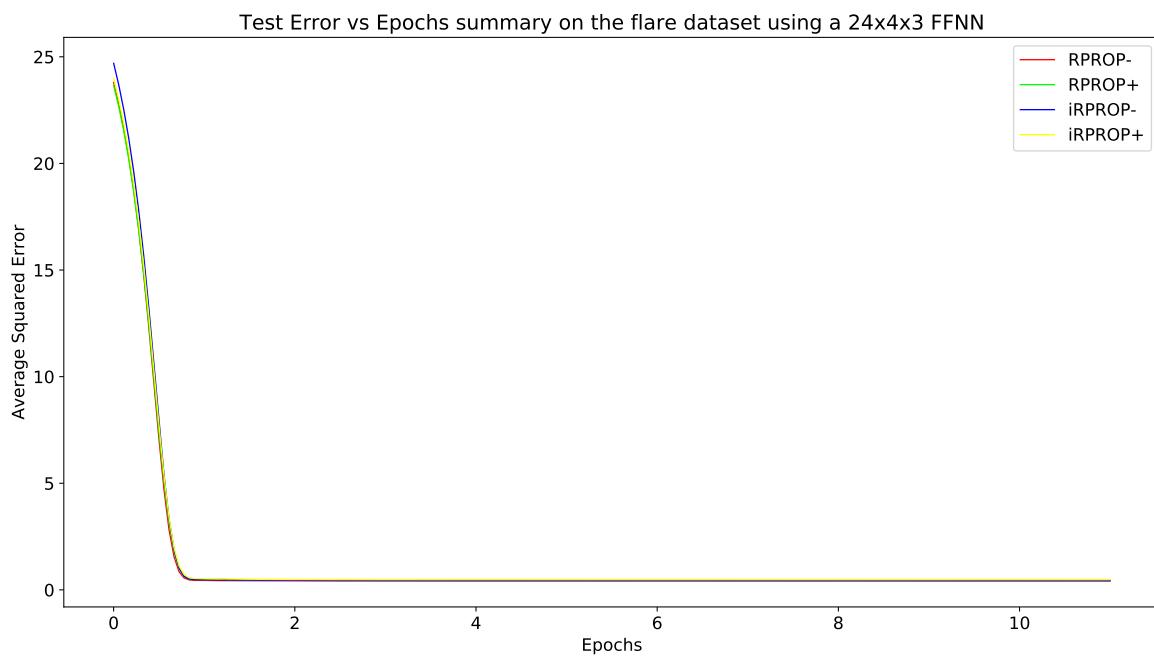
### 12.1.3 Flare dataset

This section summarises the results obtained on the flare dataset.

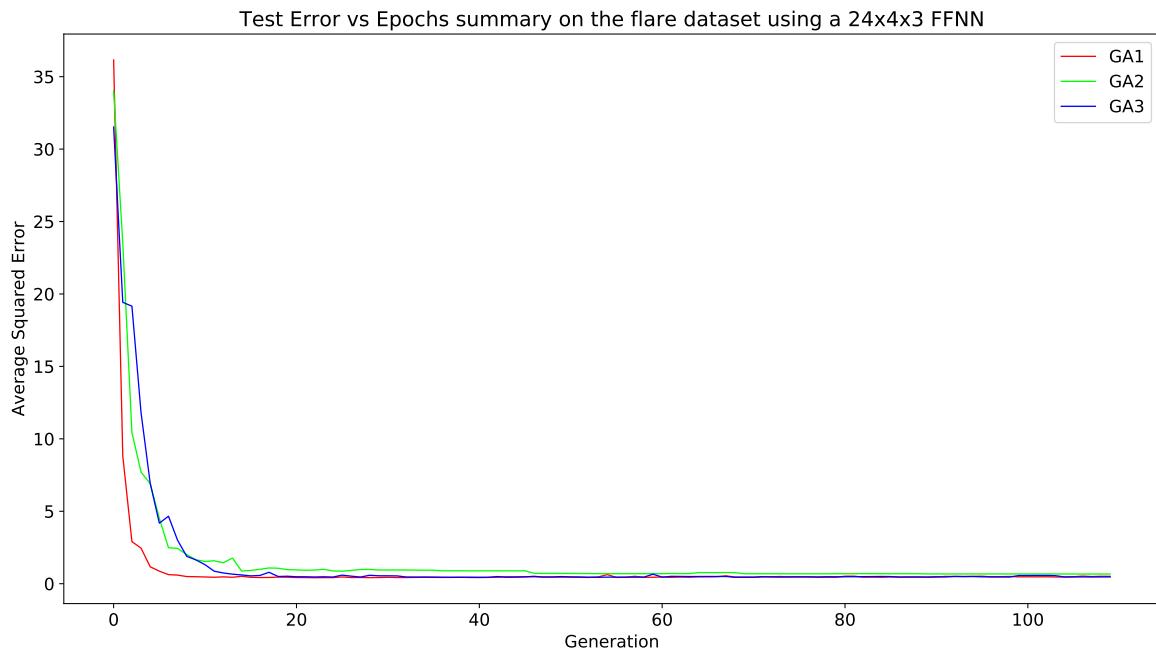
#### 24 × 4 × 3 Architecture:



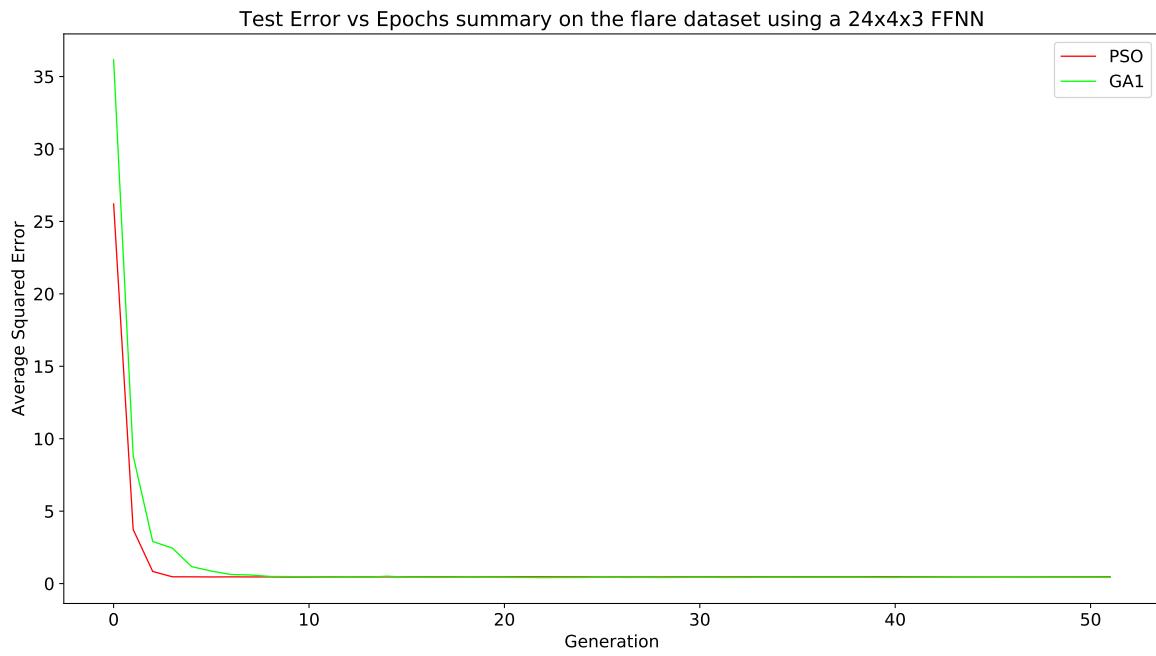
**Figure 28.** Graph of test error vs epochs for the gradient based algorithms



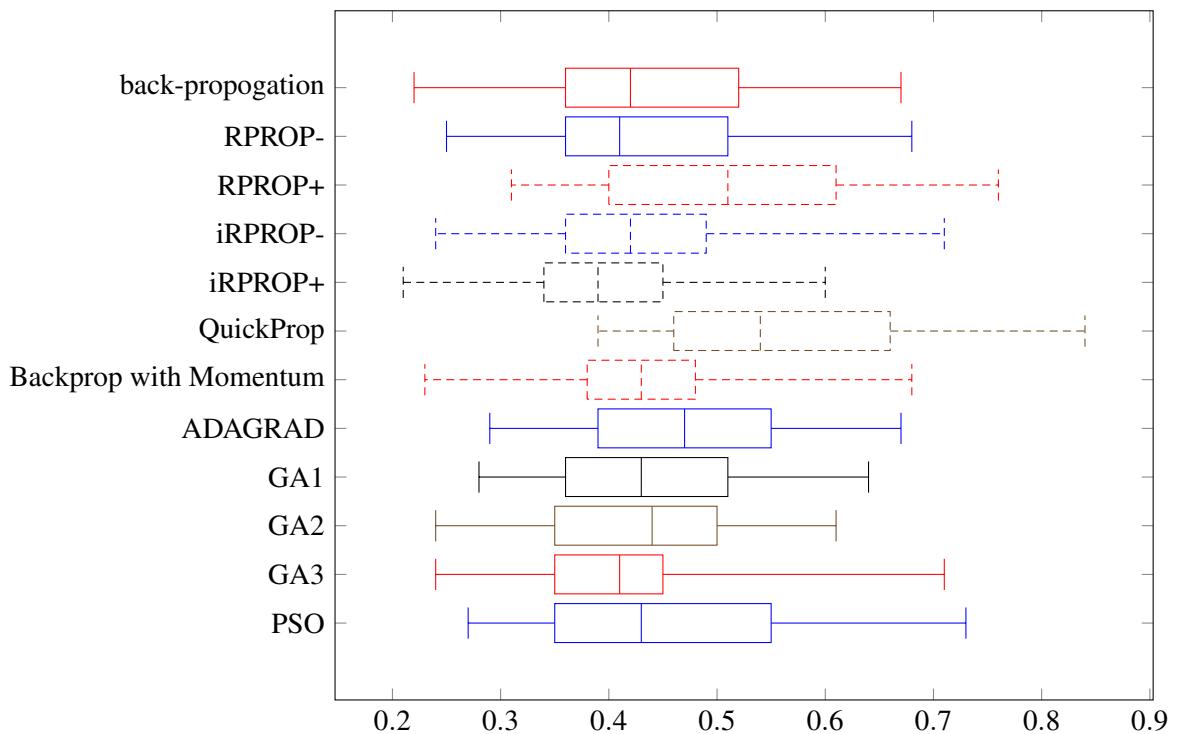
**Figure 29.** Graph of test error vs epochs for the various version of RPROP



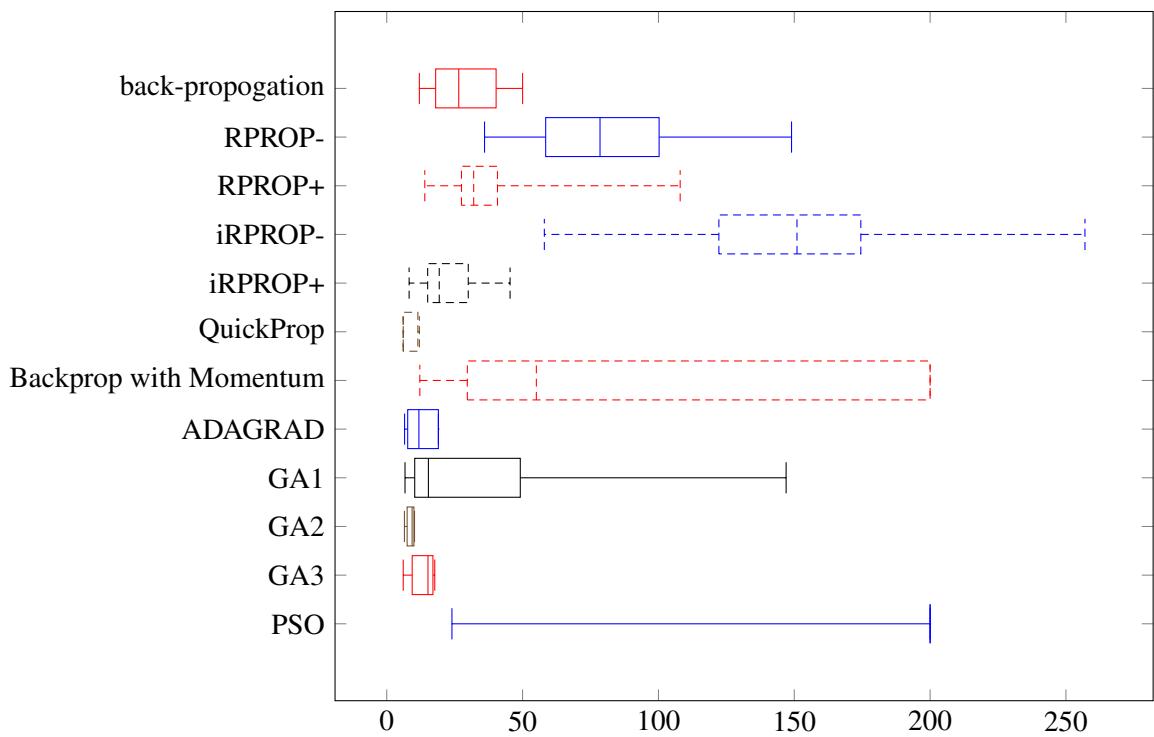
**Figure 30.** Graph of test error vs epochs for the genetic algorithms



**Figure 31.** Graph comparing GA and PSO



**Figure 32.** Best test errors achieved

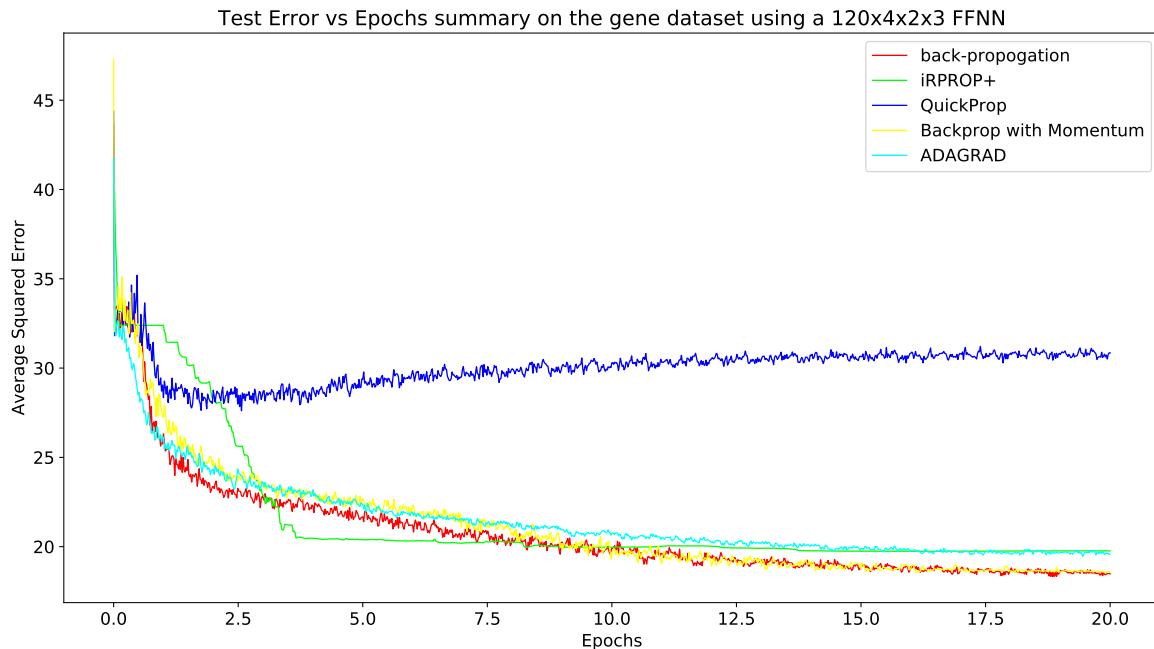


**Figure 33.** Number of epochs until algorithm terminated

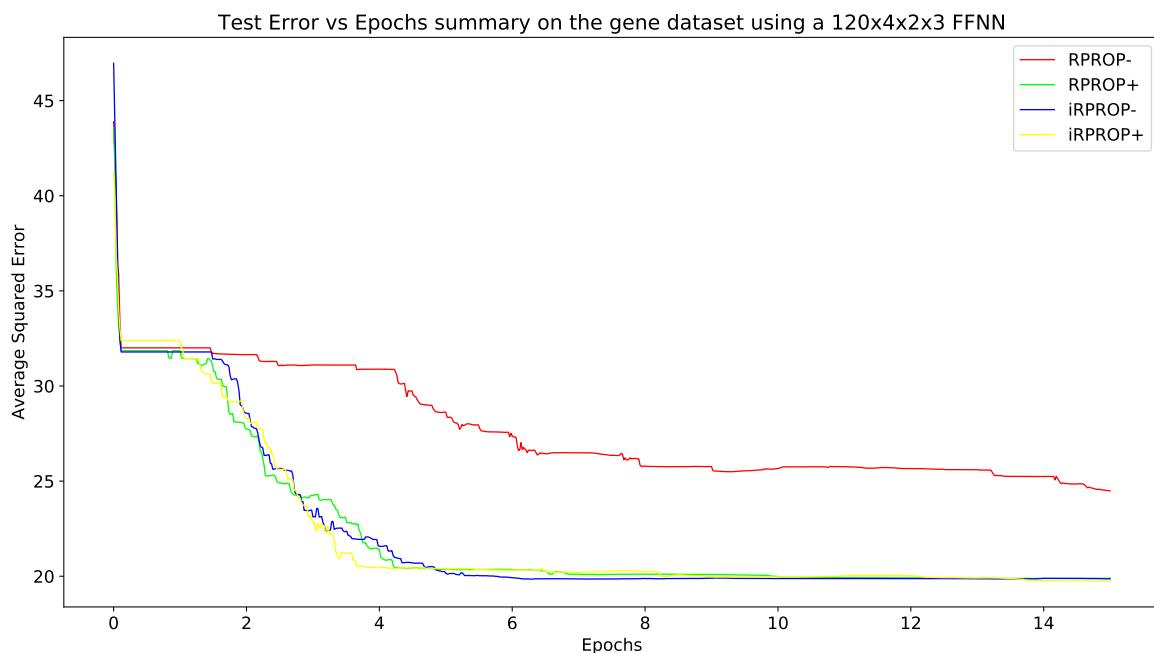
### 12.1.4 Gene dataset

This section summarises the results obtained on the gene dataset.

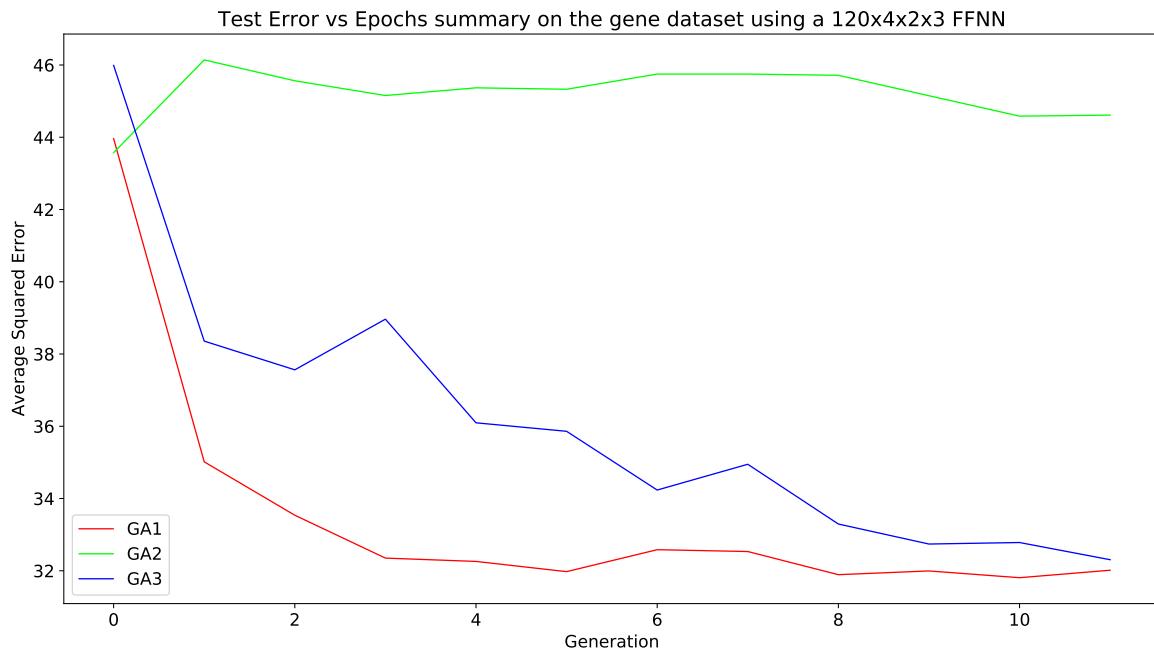
**120 × 4 × 2 × 3 Architecture:**



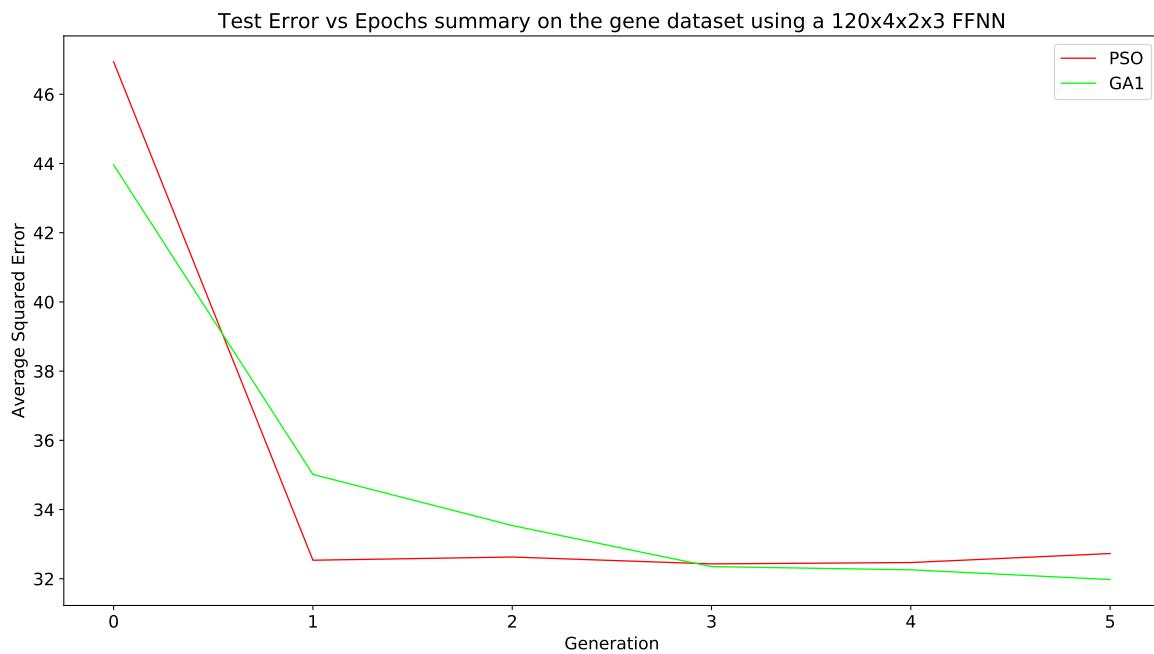
**Figure 34.** Graph of test error vs epochs for the gradient based algorithms



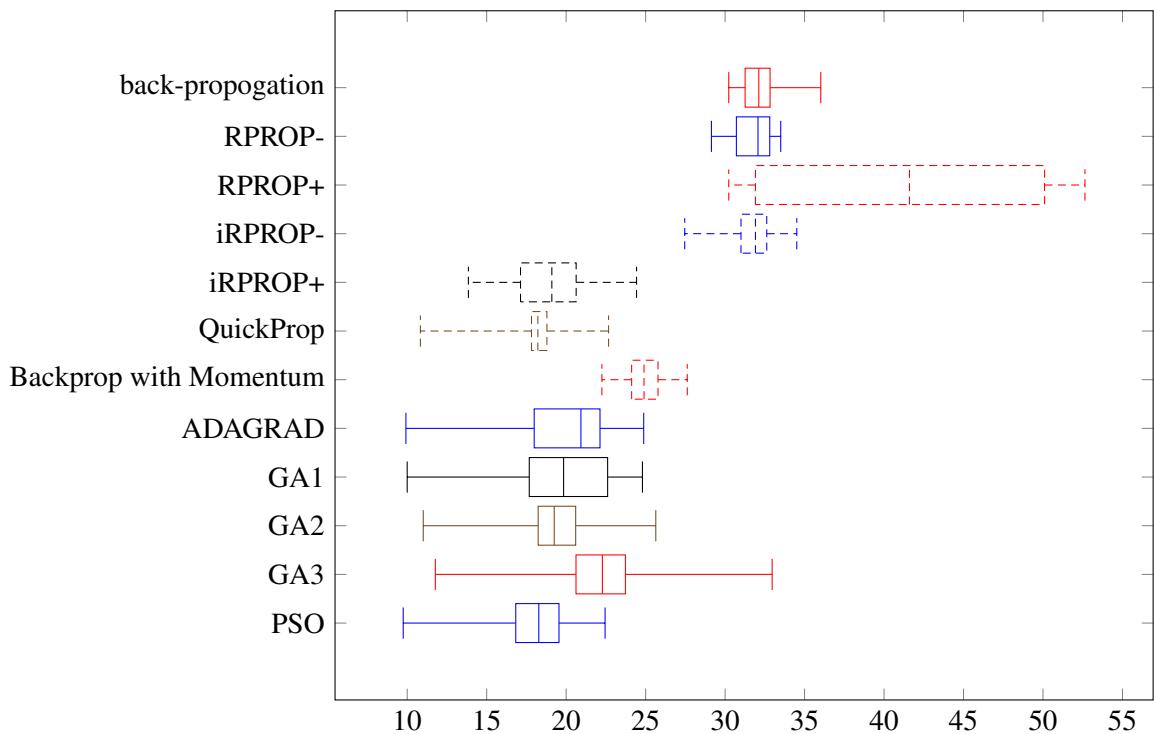
**Figure 35.** Graph of test error vs epochs for the various version of RPROP



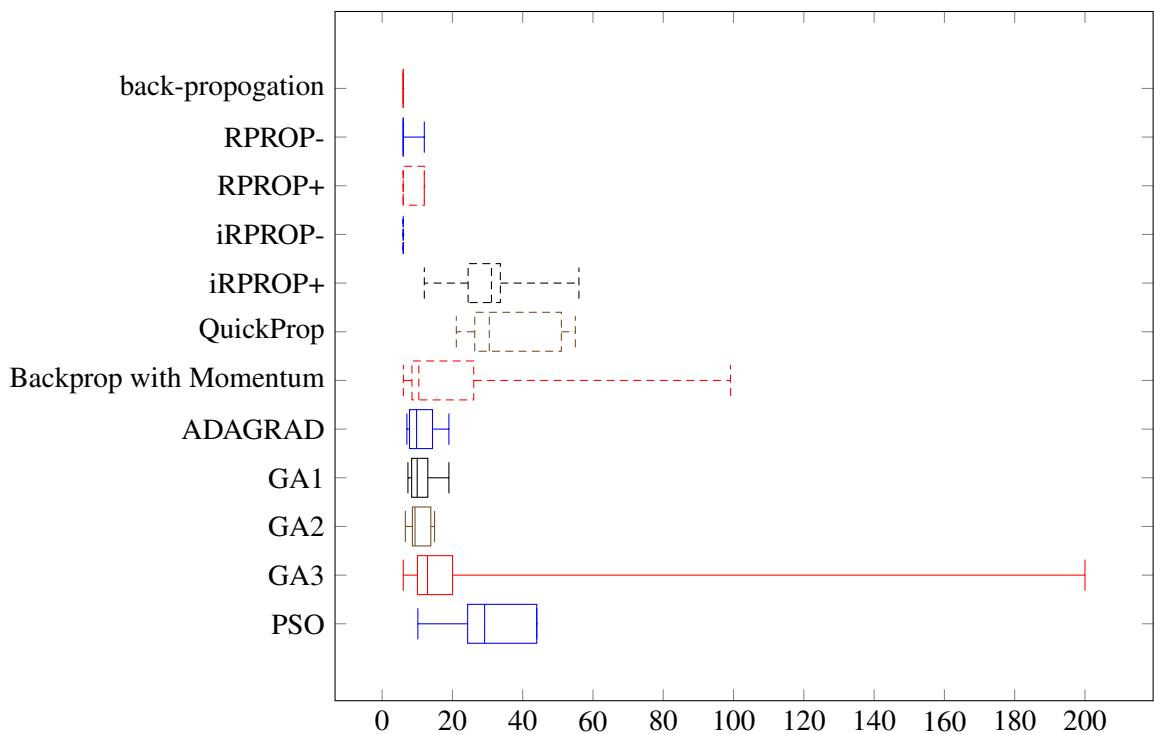
**Figure 36.** Graph of test error vs epochs for the genetic algorithms



**Figure 37.** Graph comparing GA and PSO

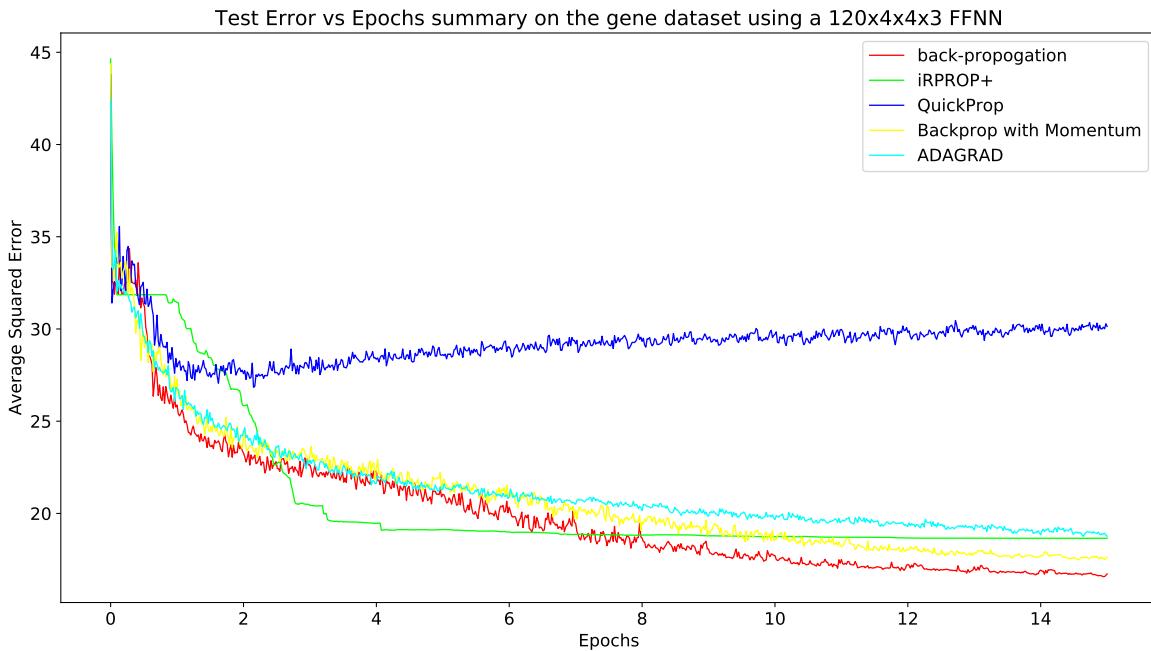


**Figure 38.** Best test errors achieved

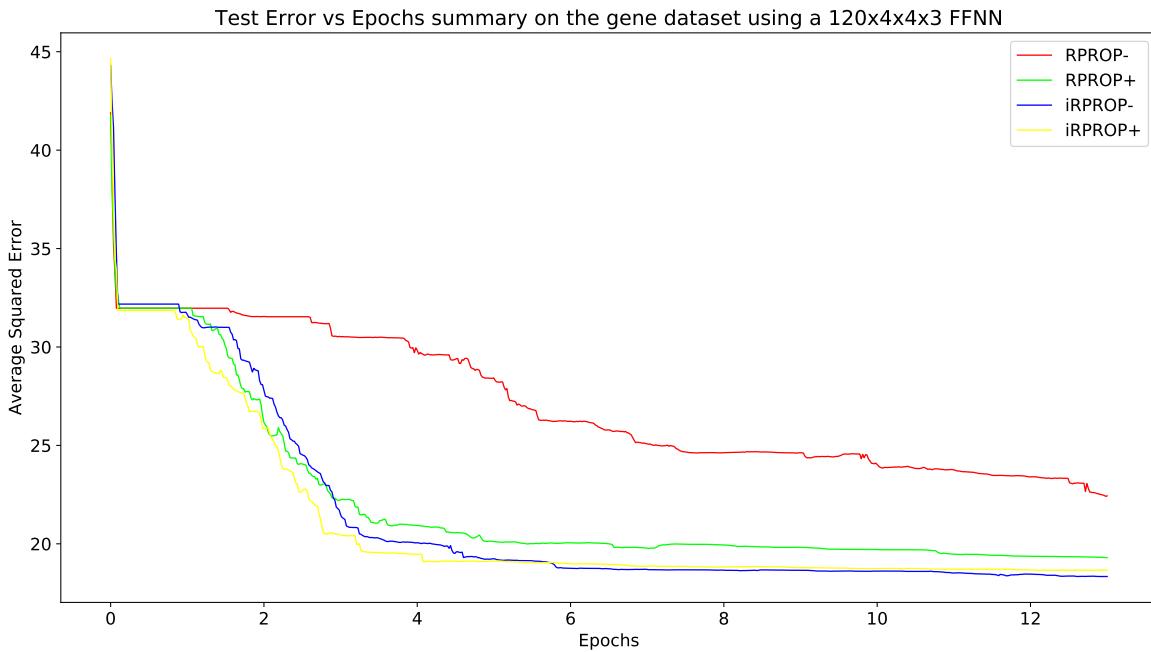


**Figure 39.** Number of epochs until algorithm terminated

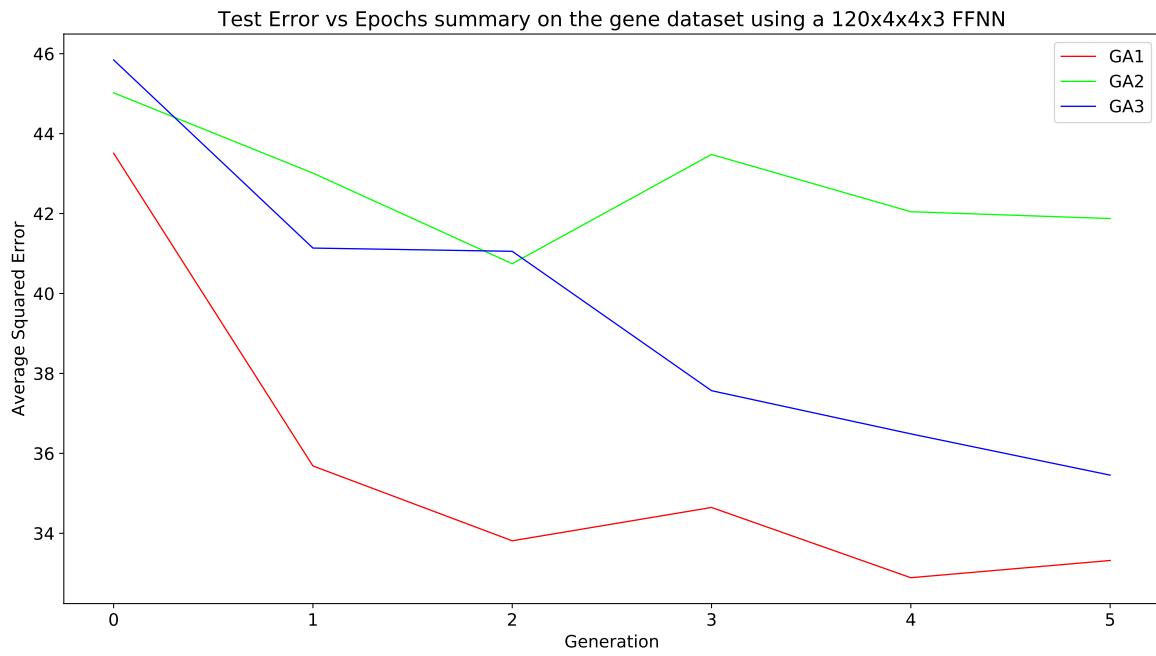
### **$120 \times 4 \times 4 \times 3$ Architecture:**



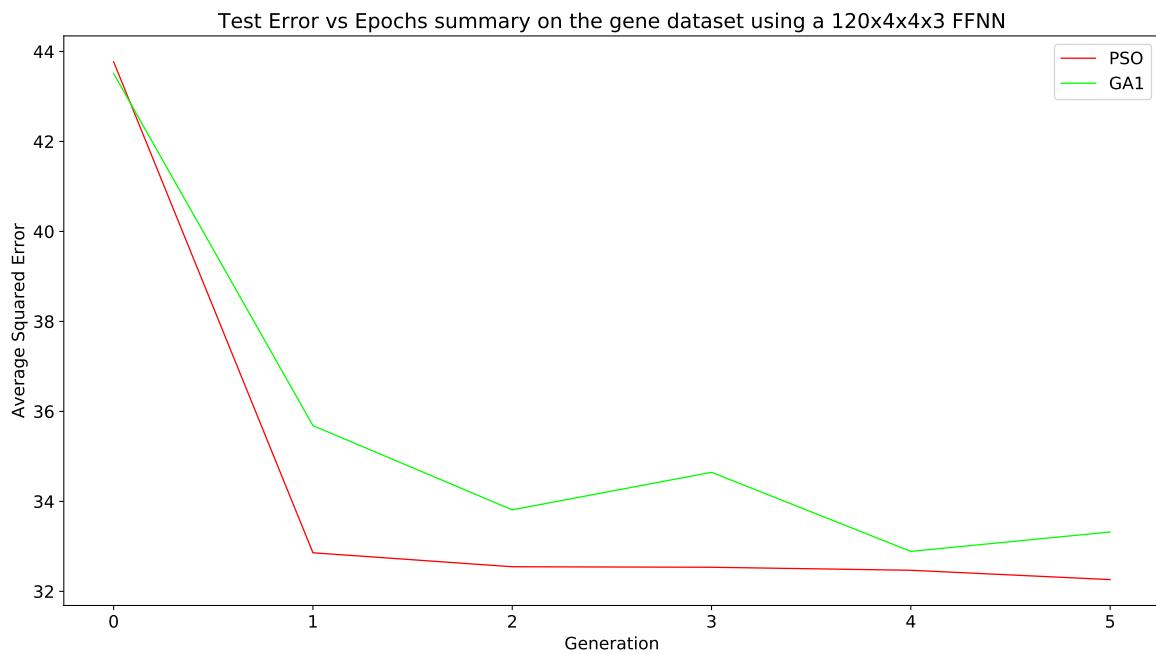
**Figure 40.** Graph of test error vs epochs for the gradient based algorithms



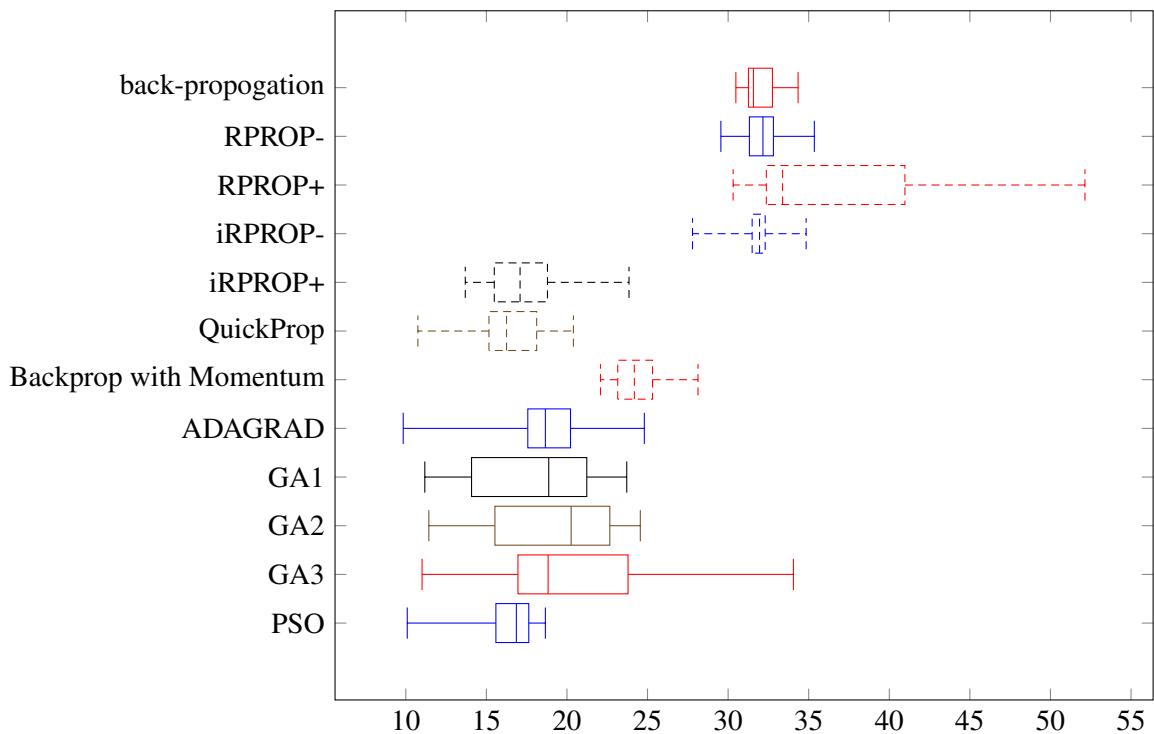
**Figure 41.** Graph of test error vs epochs for the various version of RPROP



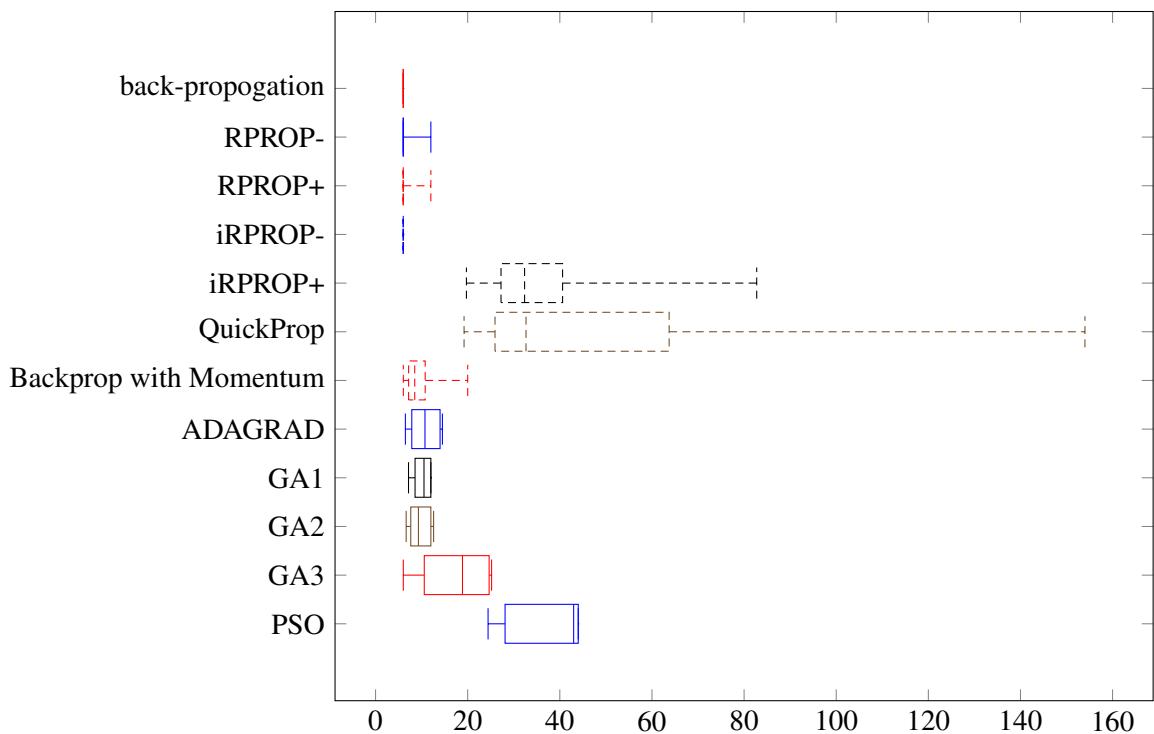
**Figure 42.** Graph of test error vs epochs for the genetic algorithms



**Figure 43.** Graph comparing GA and PSO

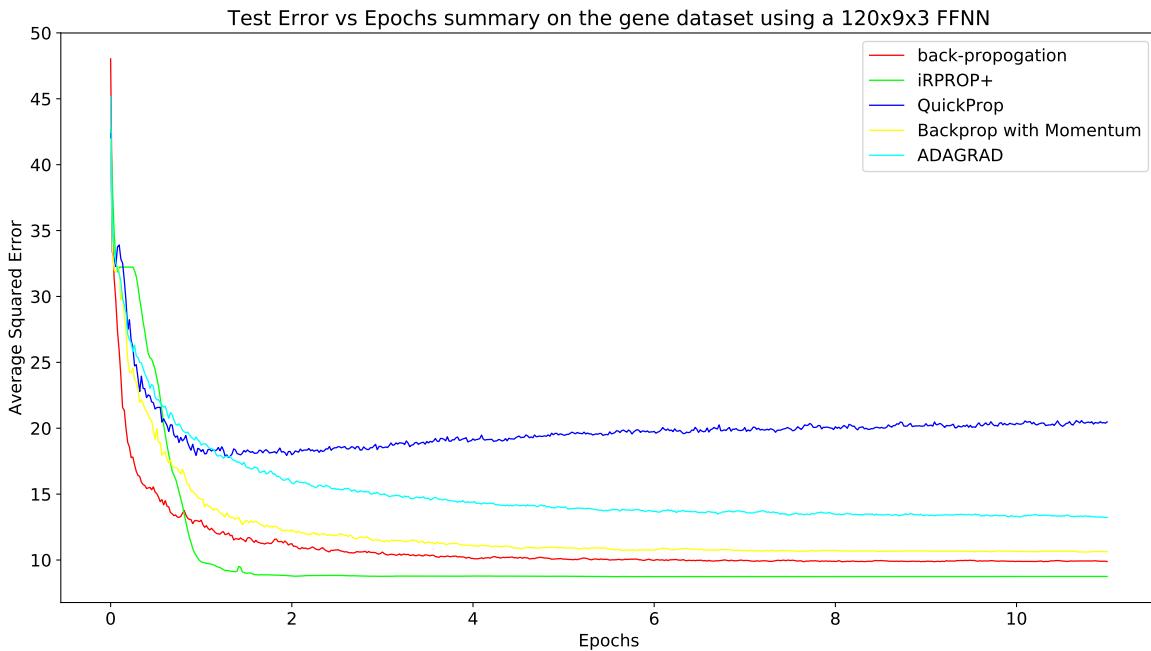


**Figure 44.** Best test errors achieved

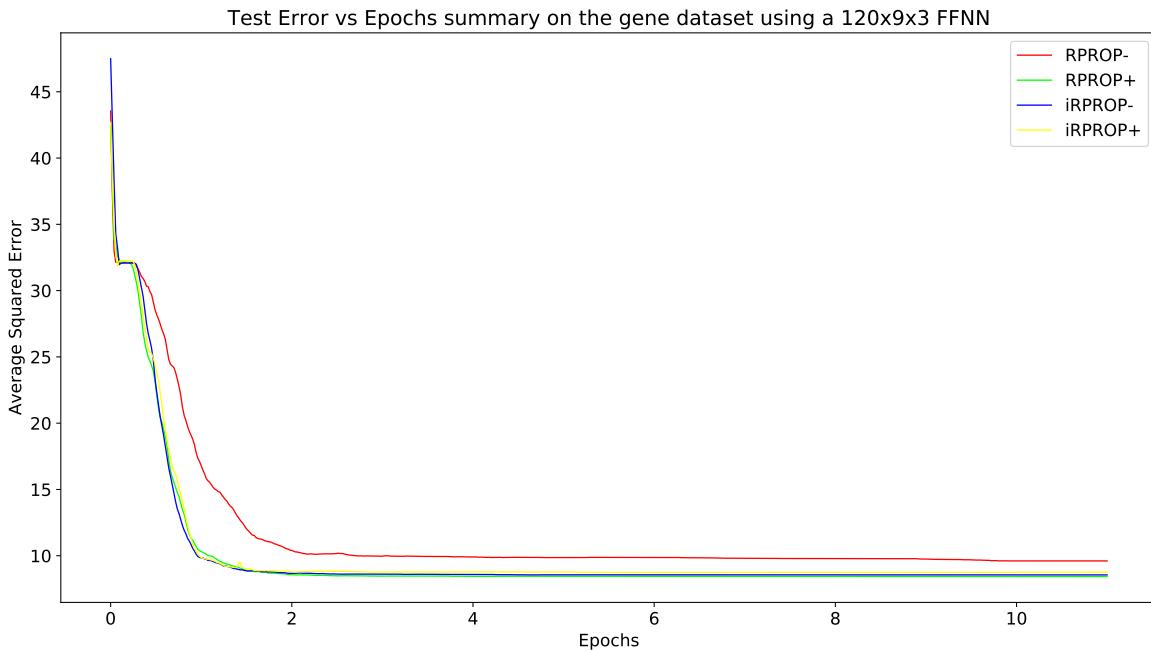


**Figure 45.** Number of epochs until algorithm terminated

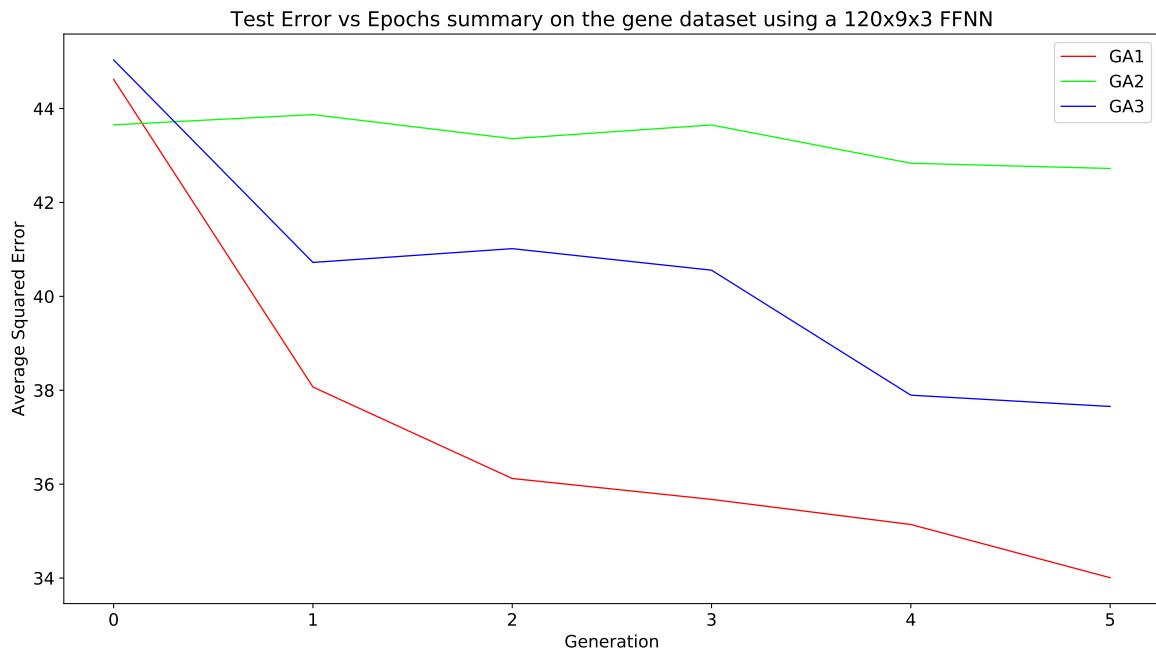
### **120 × 9 × 3 Architecture:**



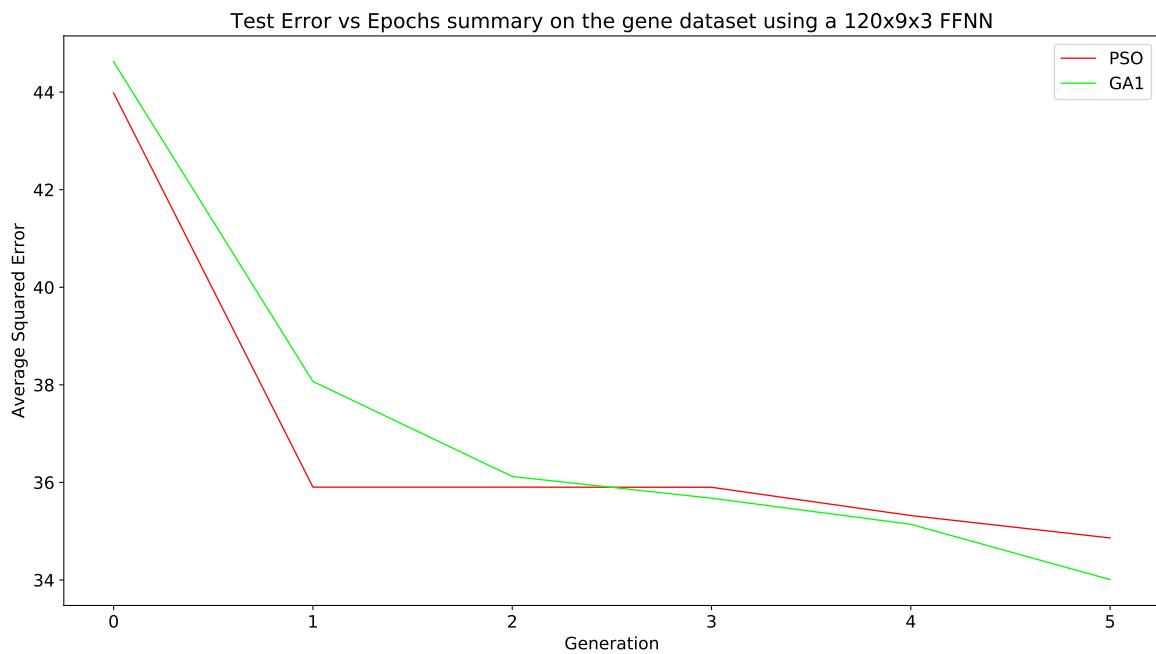
**Figure 46.** Graph of test error vs epochs for the gradient based algorithms



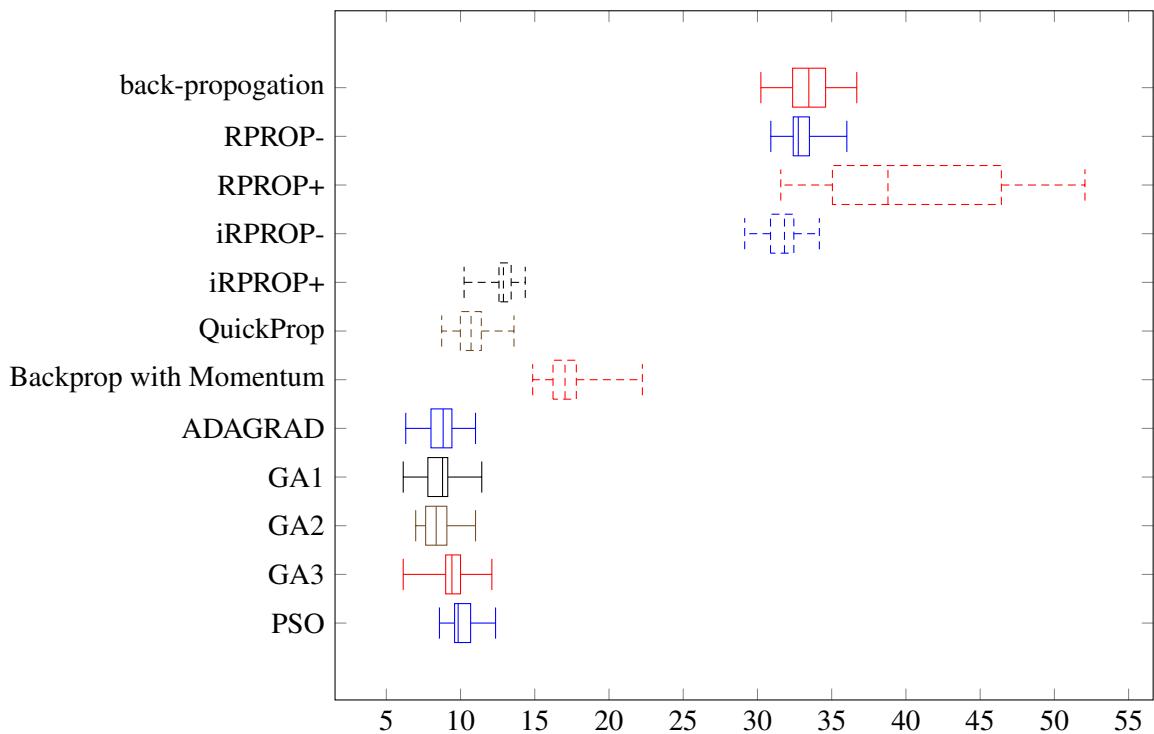
**Figure 47.** Graph of test error vs epochs for the various version of RPROP



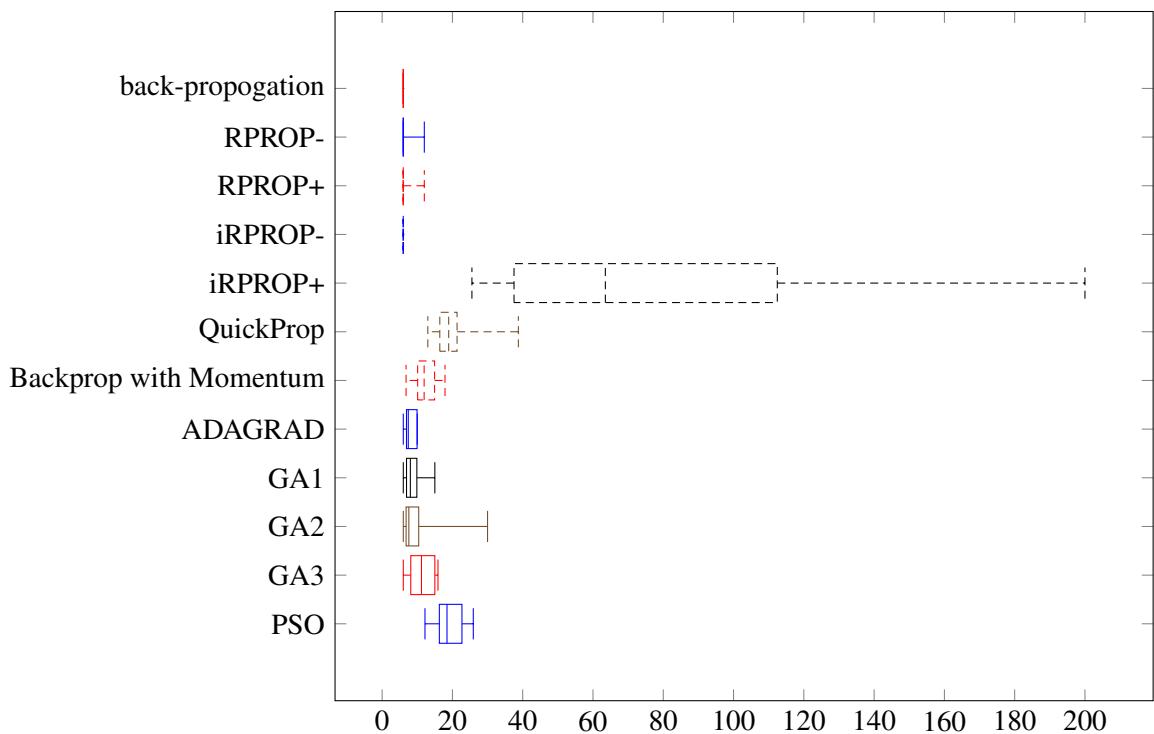
**Figure 48.** Graph of test error vs epochs for the genetic algorithms



**Figure 49.** Graph comparing GA and PSO



**Figure 50.** Best test errors achieved

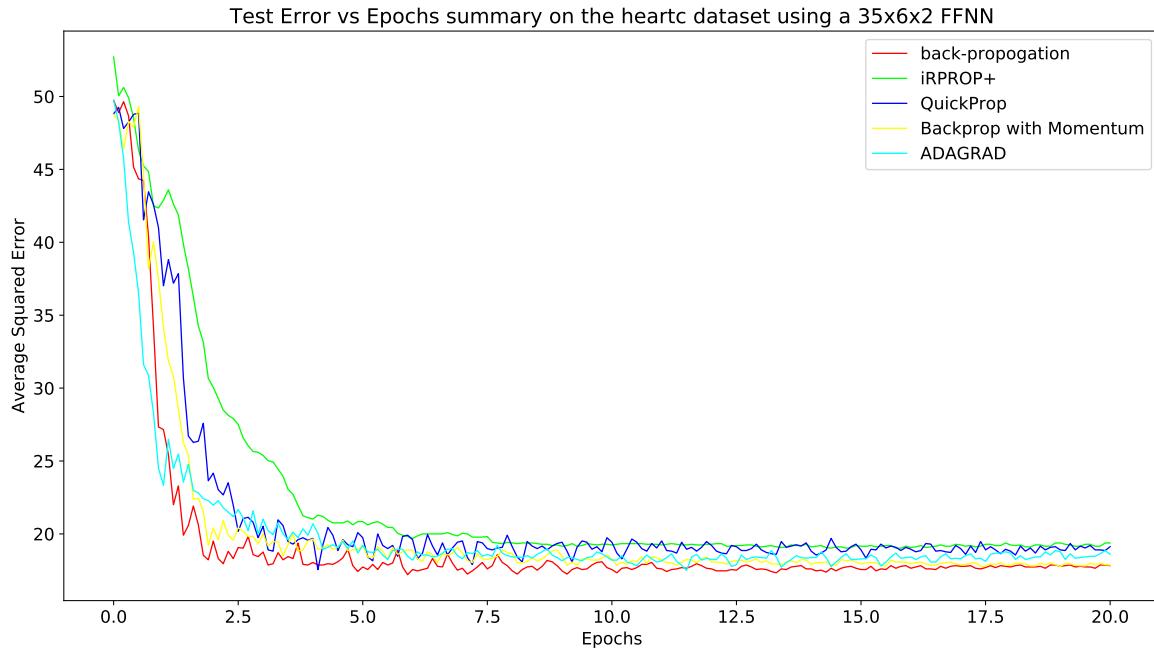


**Figure 51.** Number of epochs until algorithm terminated

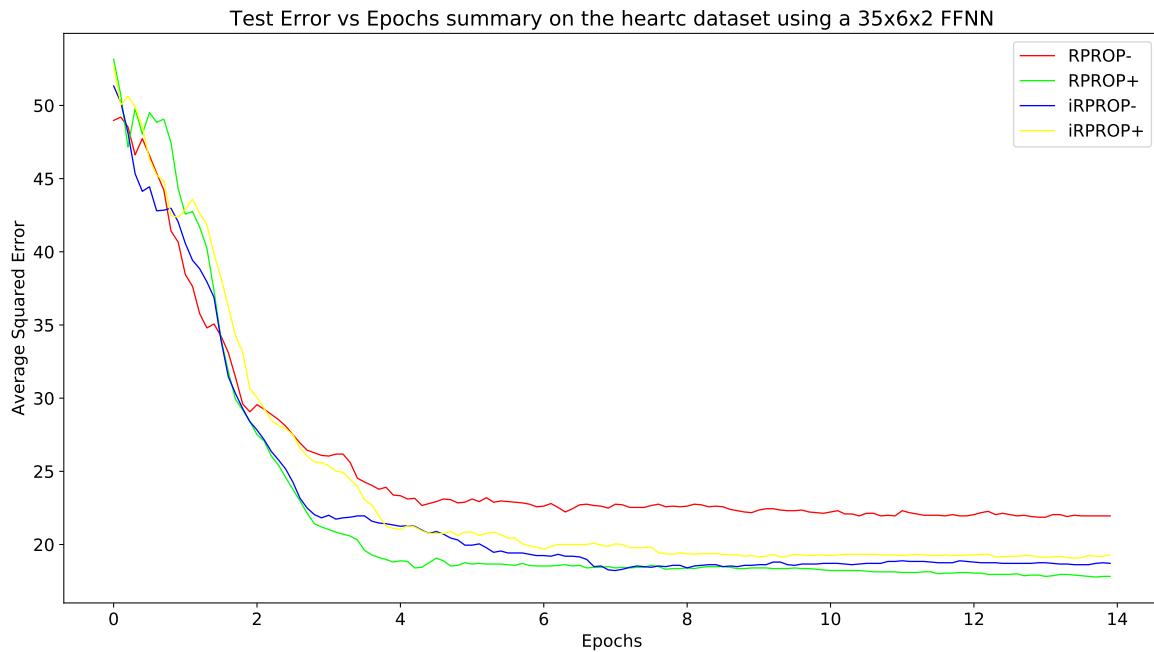
### 12.1.5 Heartc dataset

This section summarises the results obtained on the heartc dataset.

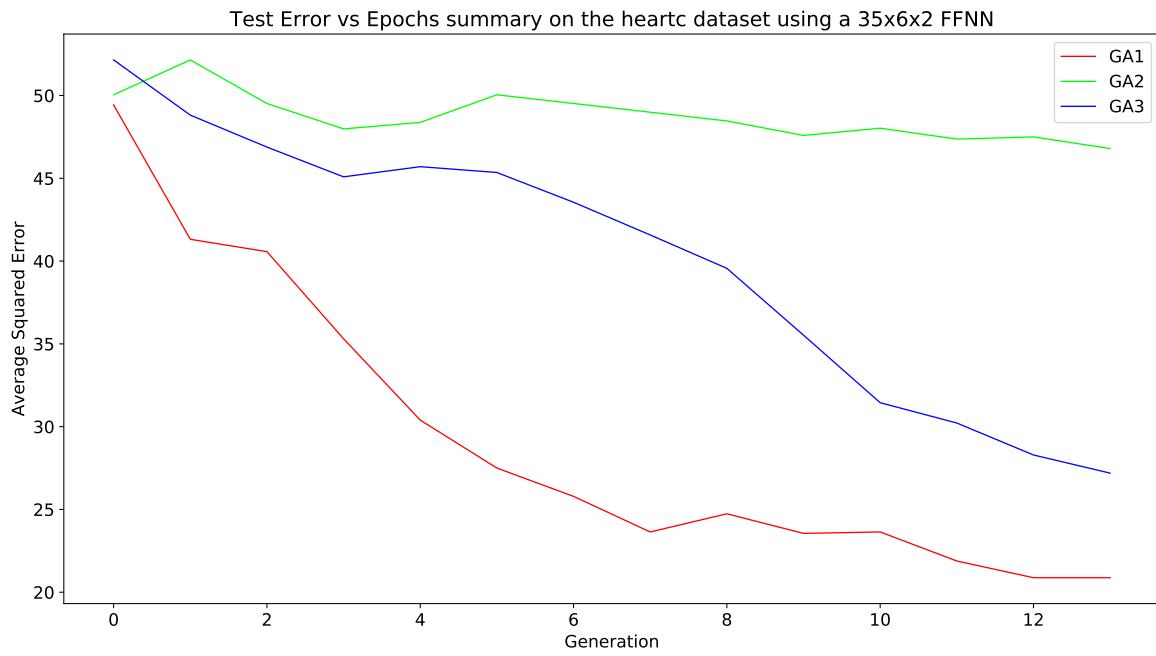
#### 35 × 6 × 2 Architecture:



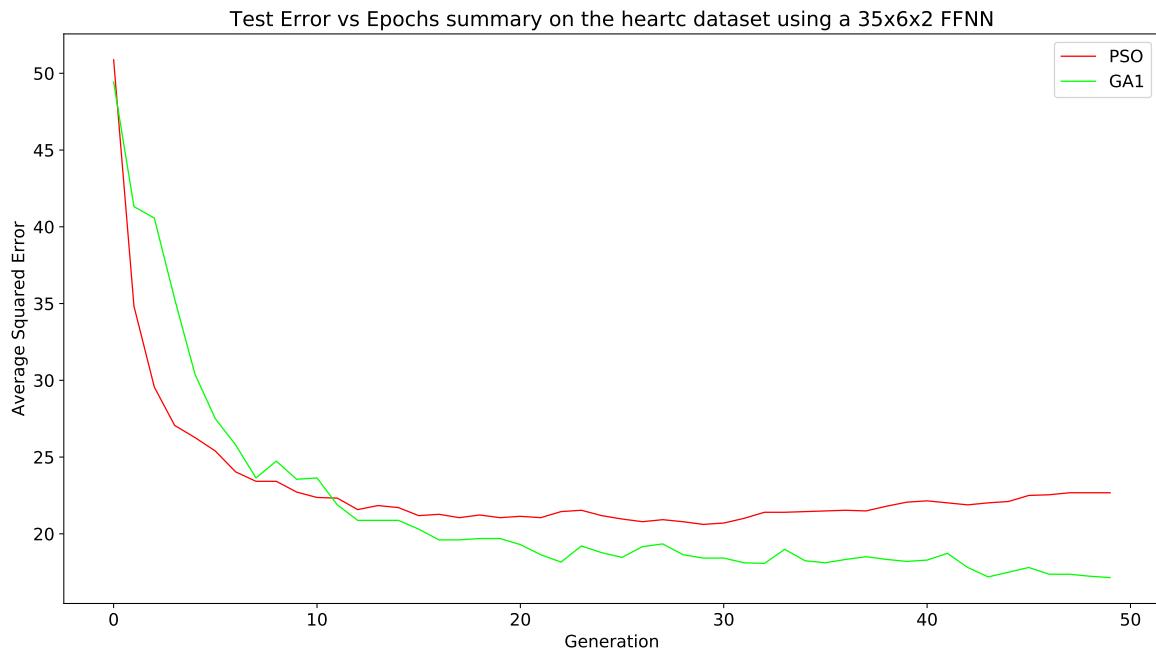
**Figure 52.** Graph of test error vs epochs for the gradient based algorithms



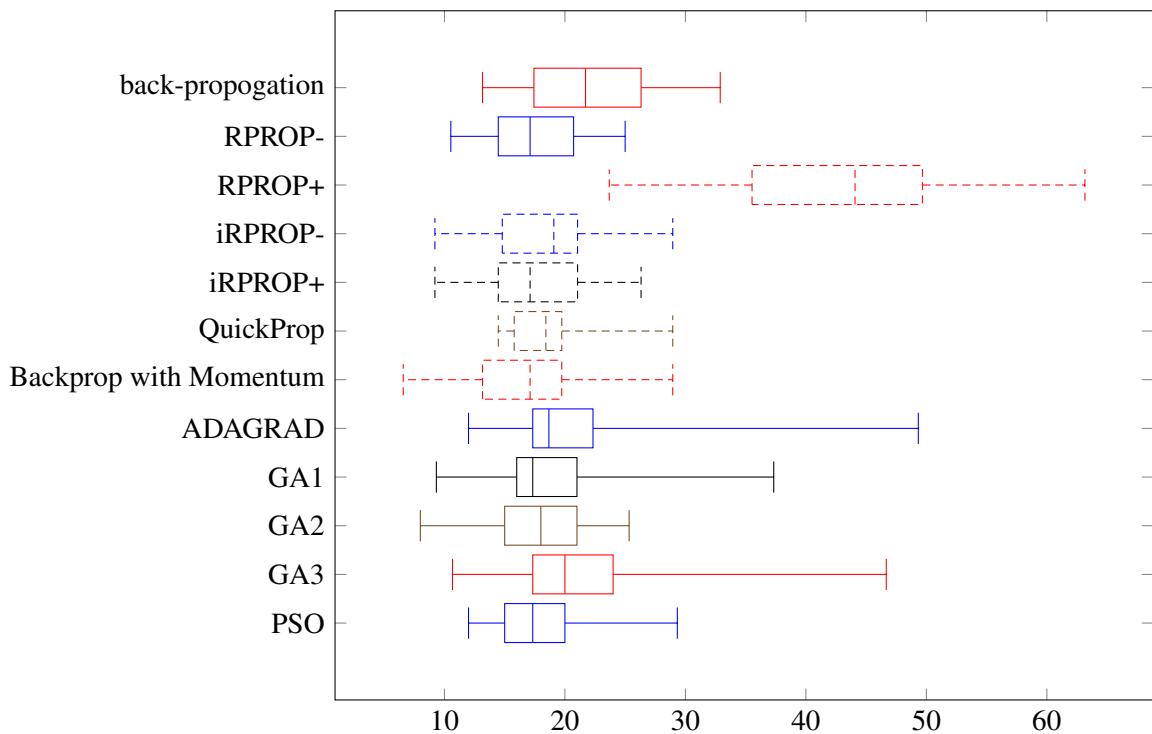
**Figure 53.** Graph of test error vs epochs for the various version of RPROP



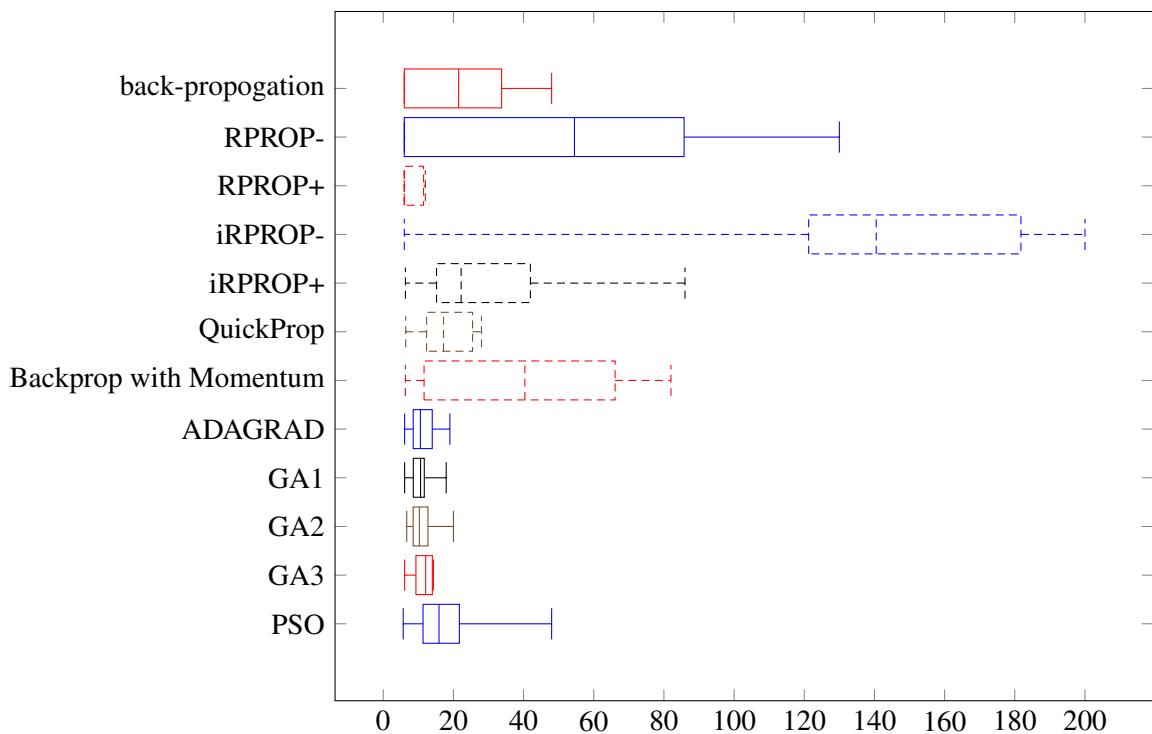
**Figure 54.** Graph of test error vs epochs for the genetic algorithms



**Figure 55.** Graph comparing GA and PSO

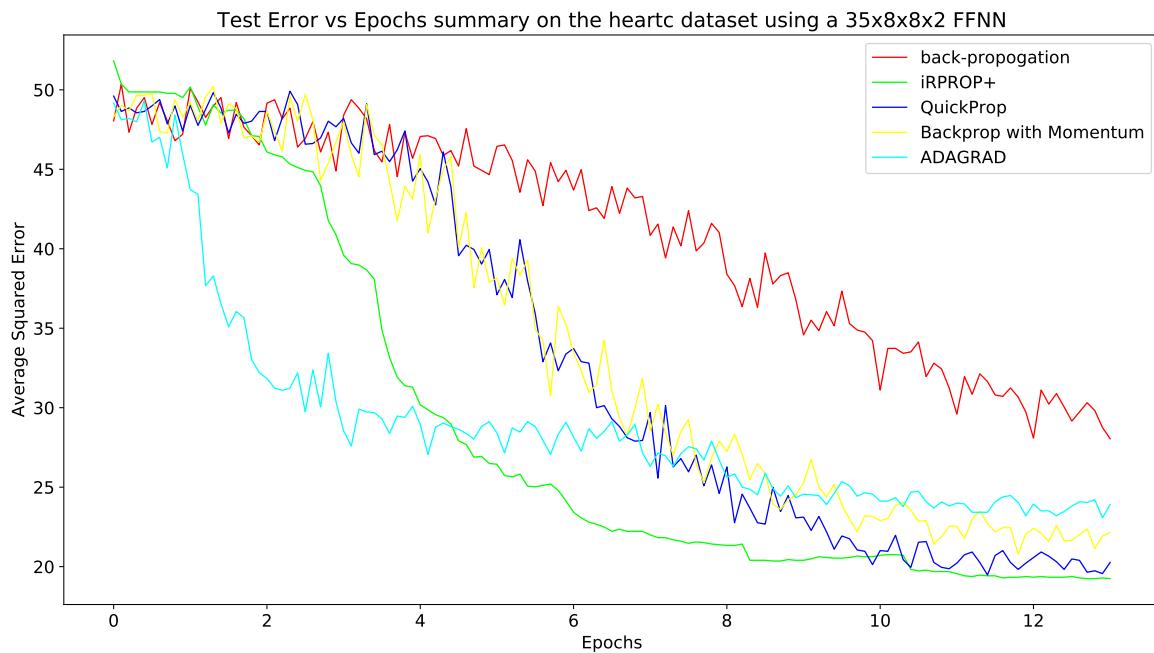


**Figure 56.** Best test errors achieved

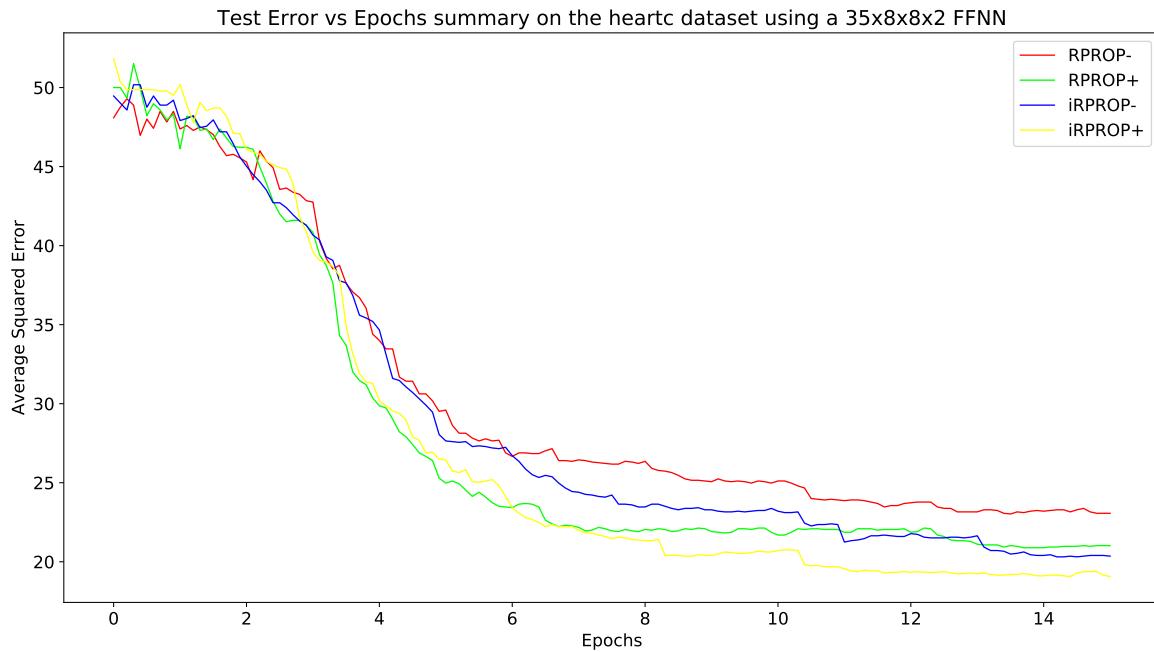


**Figure 57.** Number of epochs until algorithm terminated

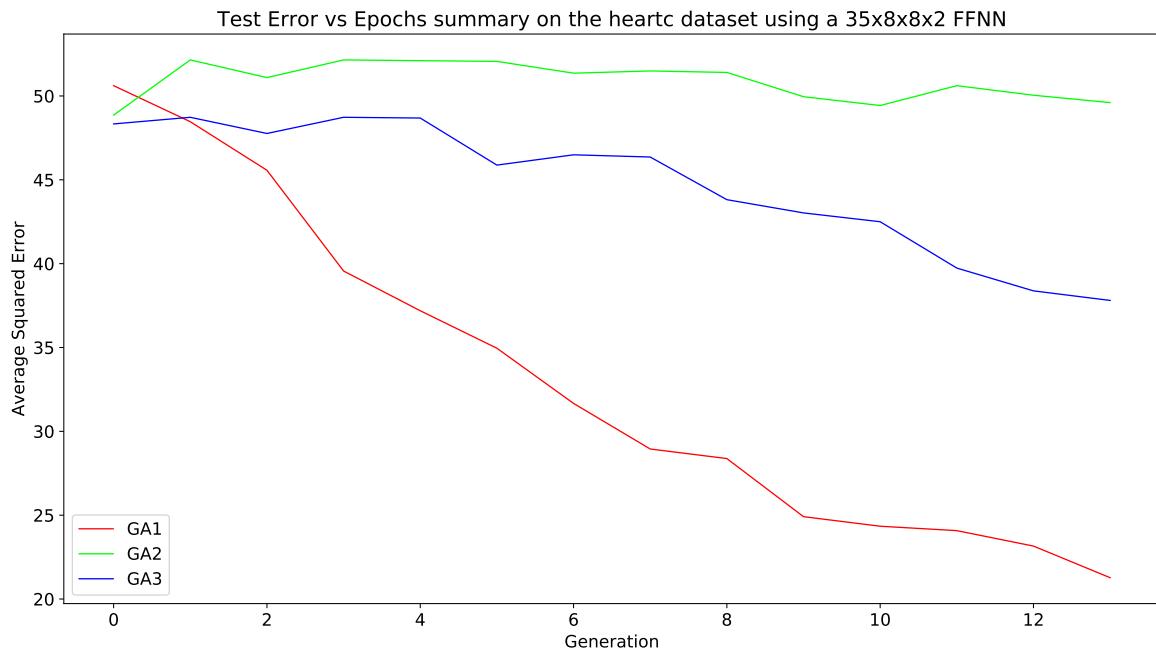
**35 × 8 × 8 × 2 Architecture:**



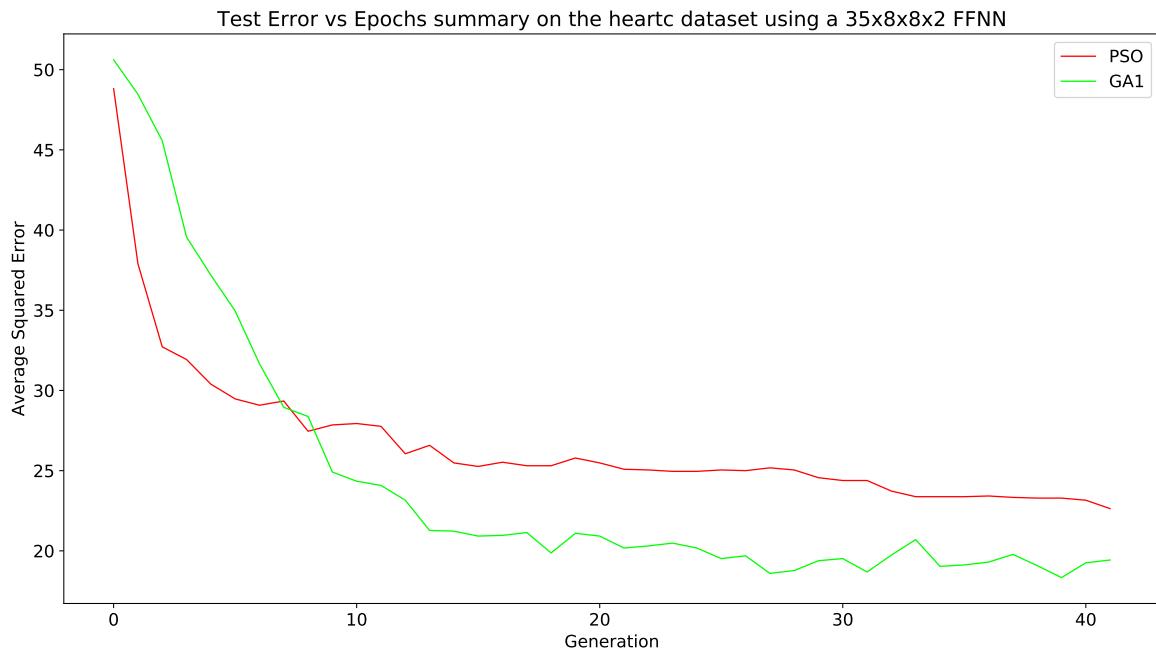
**Figure 58.** Graph of test error vs epochs for the gradient based algorithms



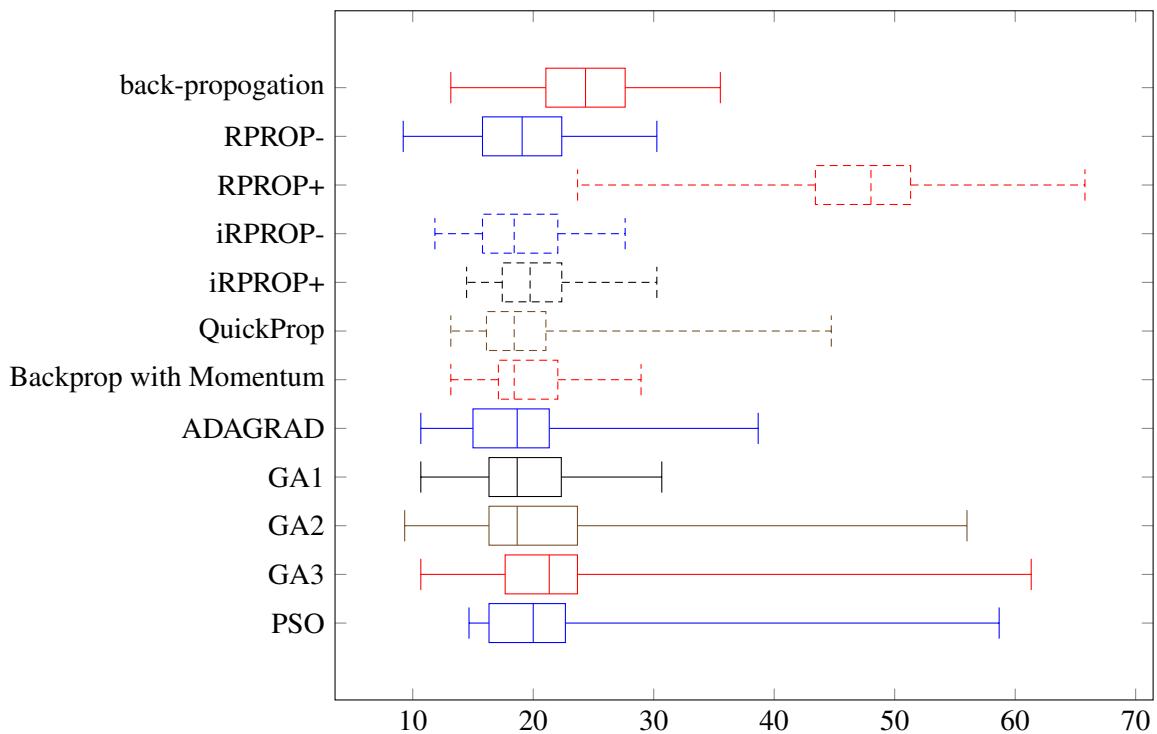
**Figure 59.** Graph of test error vs epochs for the various version of RPROP



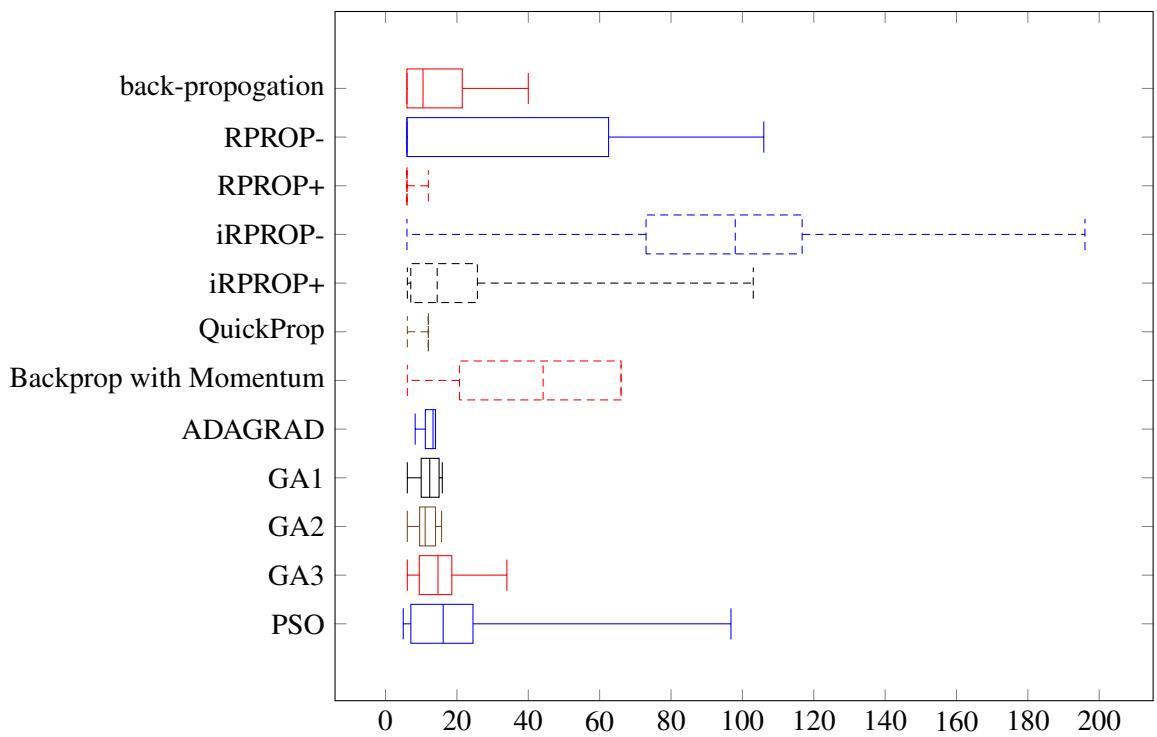
**Figure 60.** Graph of test error vs epochs for the genetic algorithms



**Figure 61.** Graph comparing GA and PSO



**Figure 62.** Best test errors achieved

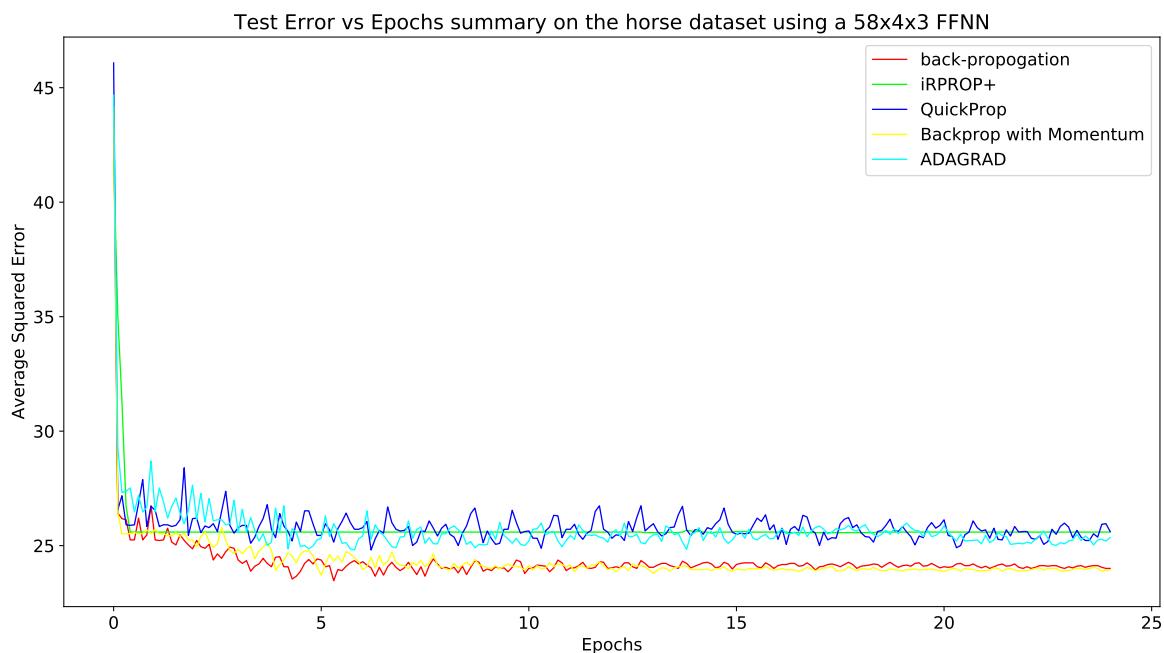


**Figure 63.** Number of epochs until algorithm terminated

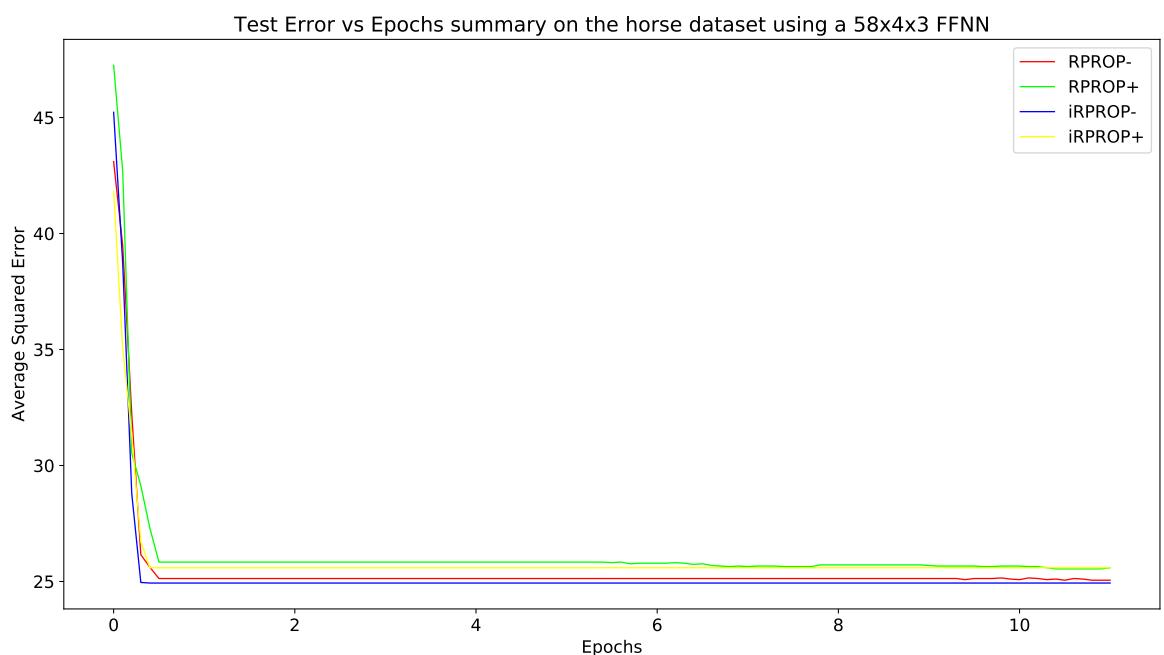
### 12.1.6 Horse dataset

This section summarises the results obtained on the horse dataset.

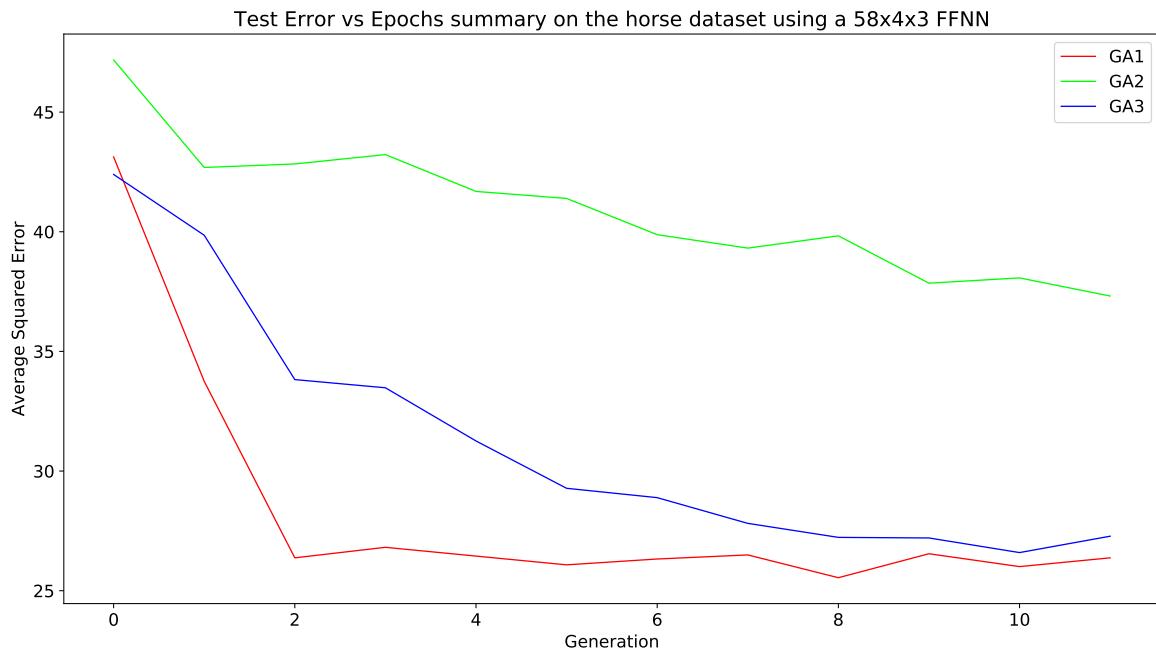
**58 × 4 × 3 Architecture:**



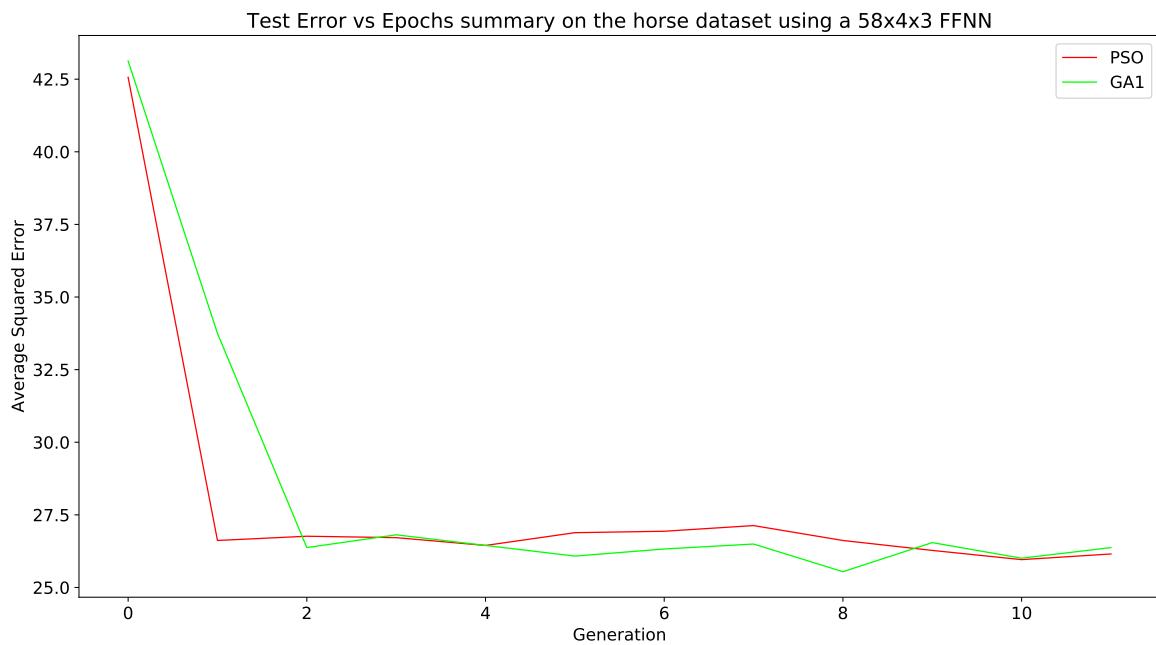
**Figure 64.** Graph of test error vs epochs for the gradient based algorithms



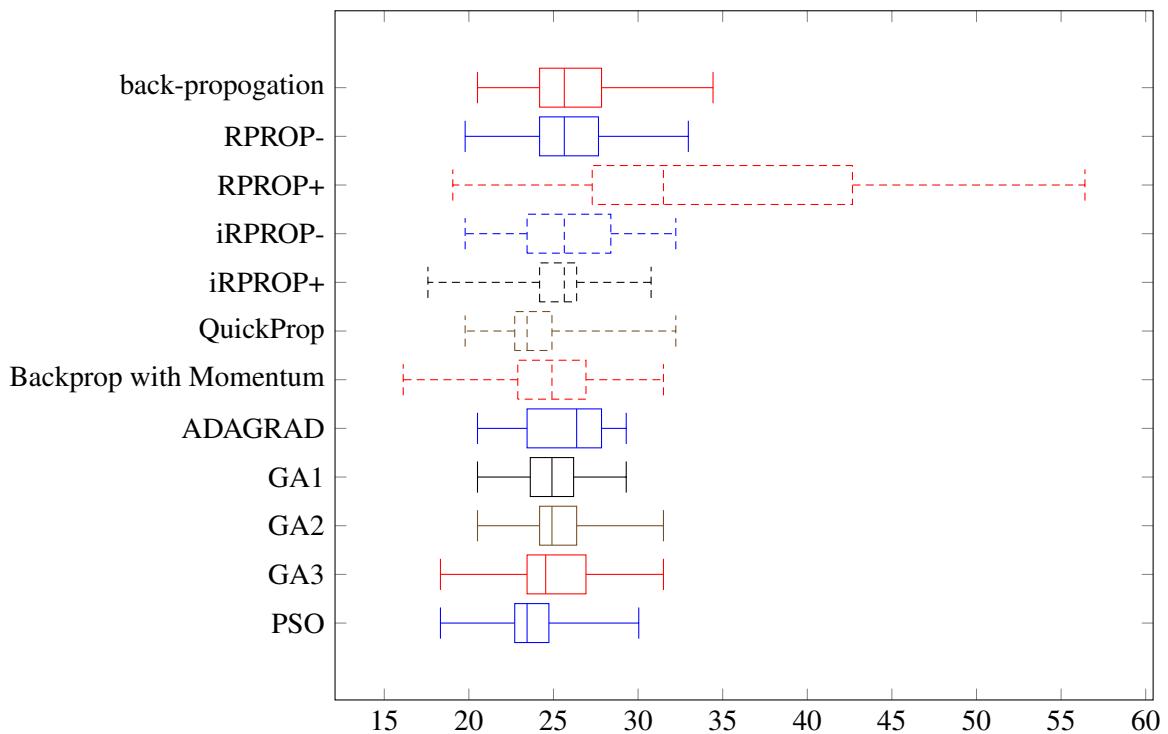
**Figure 65.** Graph of test error vs epochs for the various version of RPROP



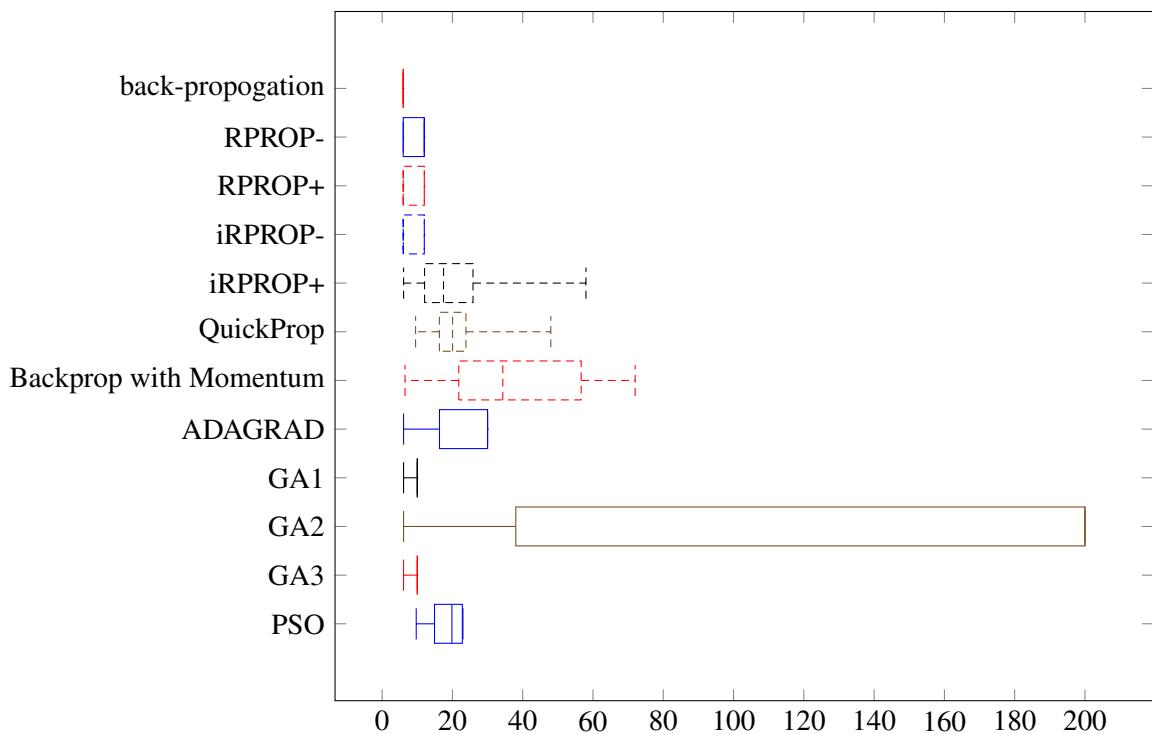
**Figure 66.** Graph of test error vs epochs for the genetic algorithms



**Figure 67.** Graph comparing GA and PSO

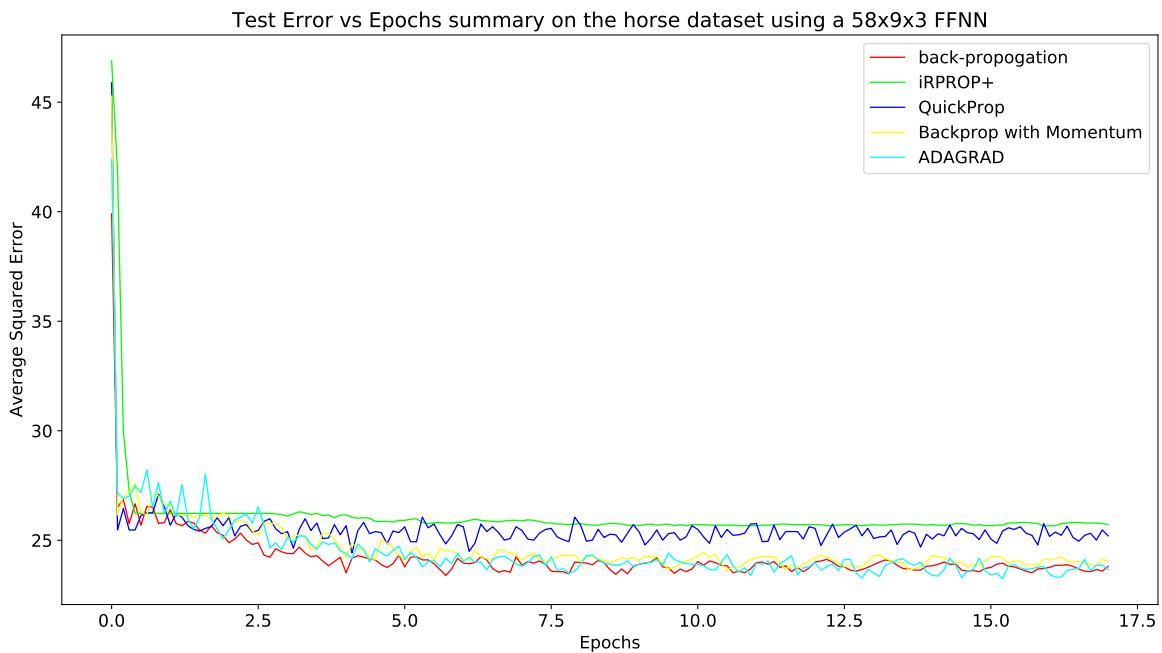


**Figure 68.** Best test errors achieved

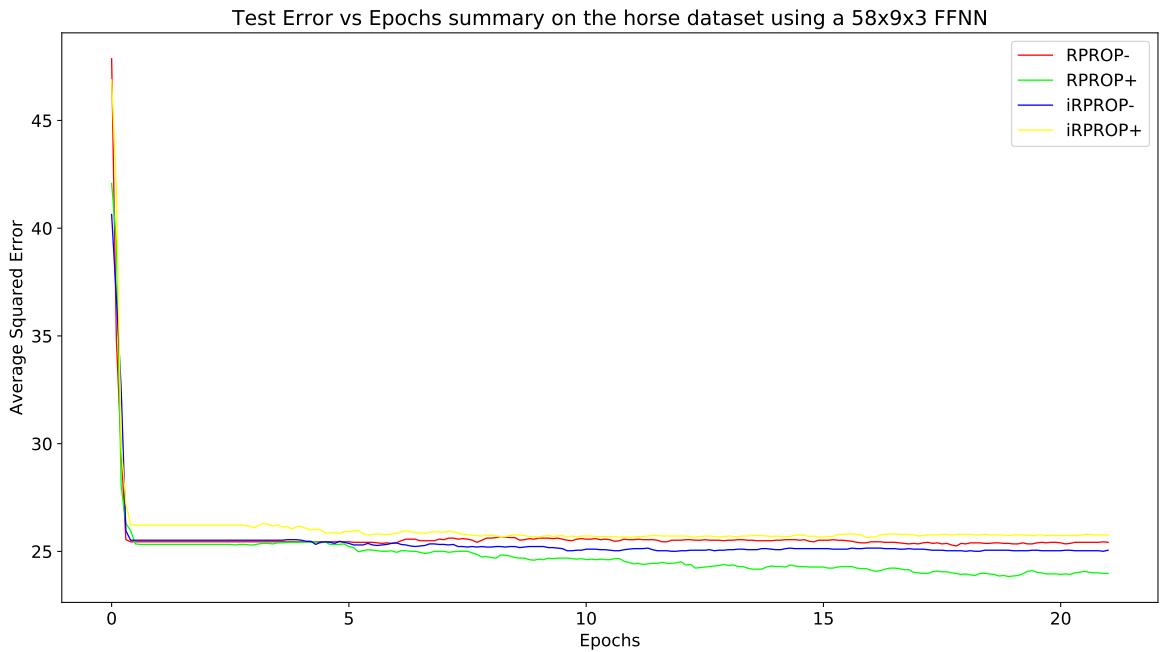


**Figure 69.** Number of epochs until algorithm terminated

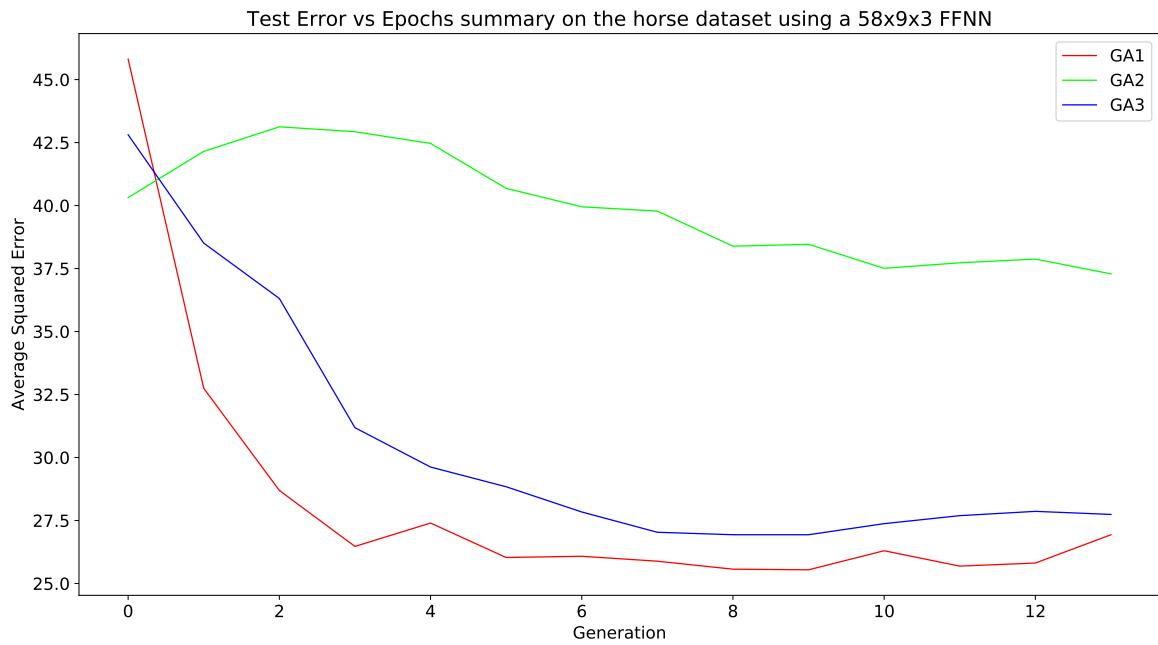
### **58 × 9 × 3 Architecture:**



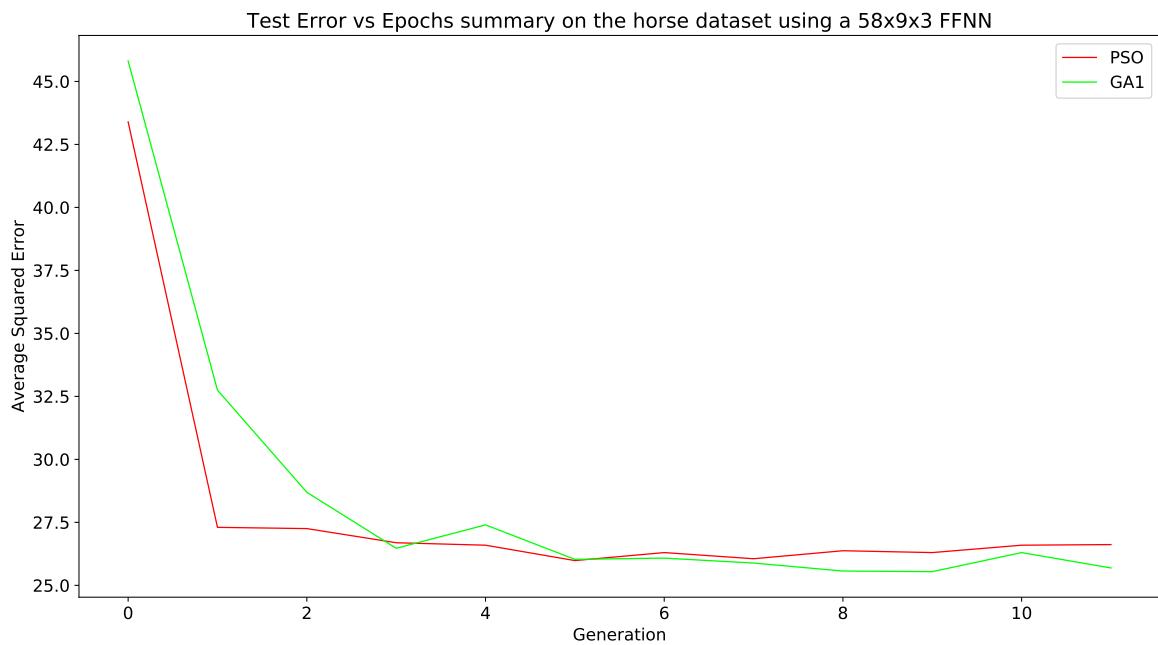
**Figure 70.** Graph of test error vs epochs for the gradient based algorithms



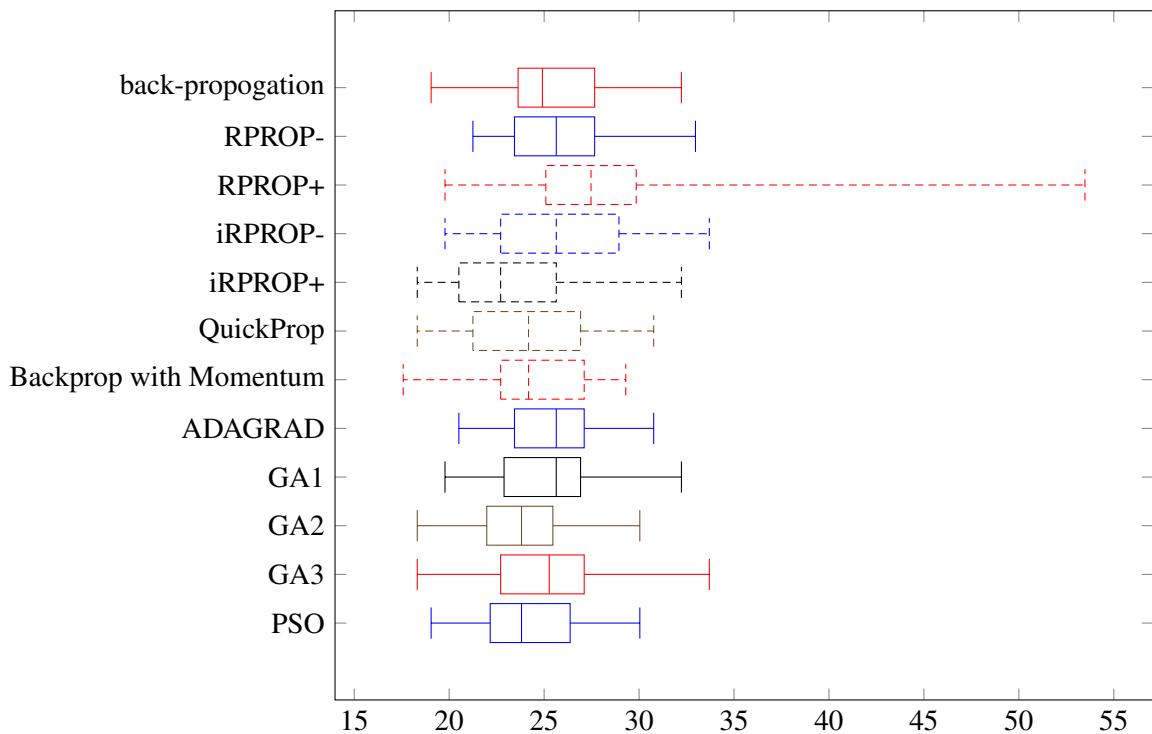
**Figure 71.** Graph of test error vs epochs for the various version of RPROP



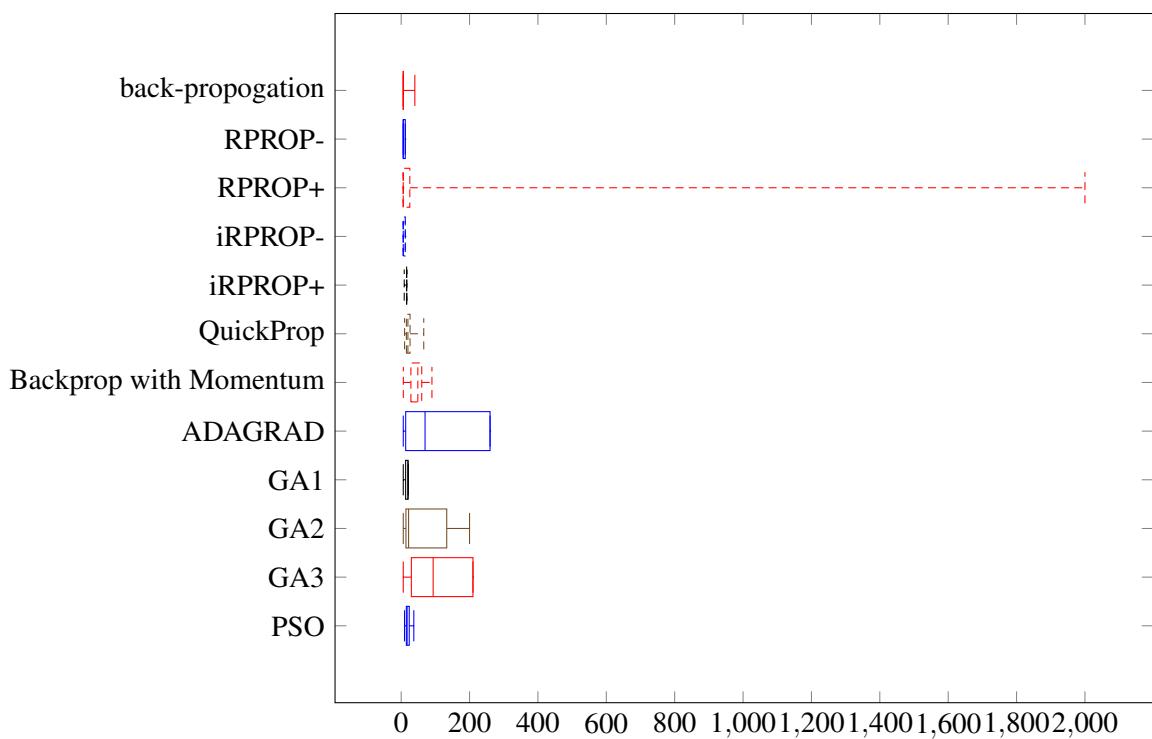
**Figure 72.** Graph of test error vs epochs for the genetic algorithms



**Figure 73.** Graph comparing GA and PSO



**Figure 74.** Best test errors achieved



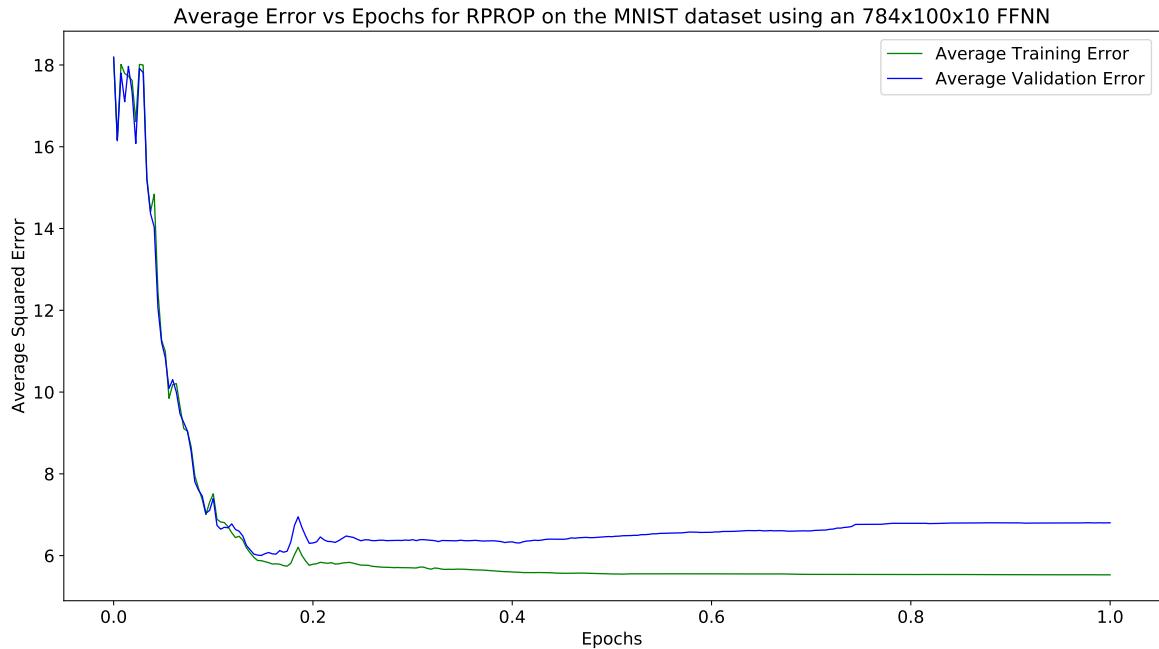
**Figure 75.** Number of epochs until algorithm terminated

## 12.2 Experiment 2: MNIST

This section shows the results for both classifiers when tested on the MNIST dataset.

### 12.2.1 Neural Network

Fig. 76 shows the training curve for the MNIST dataset.



**Figure 76.** Graph of MNIST training curve

The confusion matrix for the neural network is given in table 3 and the overall accuracy is determined to be 70.19%.

		Predicted values									
		0	1	2	3	4	5	6	7	8	9
Actual values	0	802	1	44	14	4	98	9	3	4	1
	1	1	1002	20	17	2	2	6	27	58	0
	2	39	14	744	44	44	13	40	26	62	6
	3	11	4	63	801	13	35	5	29	42	7
	4	7	3	17	17	735	11	17	15	28	132
	5	24	5	69	205	40	421	24	17	74	13
	6	26	4	138	1	34	16	722	2	15	0
	7	7	23	22	10	33	4	0	866	21	42
	8	8	30	50	148	41	116	33	19	506	23
	9	15	11	5	43	337	32	4	91	51	420

**Table 3.** Confusion matrix for the neural network

### 12.2.2 Naive Bayesian classifier

The confusion matrix for the Naive Bayesian classifier is given in table 4 and the overall accuracy is determined to be 89.97%.

		Predicted values									
		0	1	2	3	4	5	6	7	8	9
Actual values	0	915	2	13	8	0	4	15	0	20	3
	1	0	1075	19	2	13	1	0	5	19	1
	2	9	1	956	29	3	2	0	4	27	1
	3	0	2	26	925	2	23	1	7	16	8
	4	0	7	27	0	892	2	10	8	5	31
	5	7	0	7	50	3	777	10	2	32	4
	6	10	10	7	1	9	26	881	0	14	0
	7	0	6	35	10	32	0	0	842	10	93
	8	30	9	23	26	9	16	2	11	828	20
	9	9	9	17	10	8	8	0	26	16	906

**Table 4.** Confusion matrix for the Naive Bayesian classifier

## 13 Discussion

In most of the test cases, the backward-propagation performed exceptionally well. Often converging really fast and consistently getting good results. However, these are small test cases and hence it is easy for backward propagation to achieve good results. The RPROP algorithms appear to have similar performance, although iRPROP+ appears to consistently find the best result the fastest. As opposed to RPROP- which occasionally has some difficulties. These algorithms are well suited for large neural nets as they are able to run fast and converge at a rapid pace. They are also fairly stable and require less "tweaking" than the other algorithms considered in this paper. Out of all the gradient based approaches, Quickprop was the least stable and would often diverge. This can be seen by the large range in the epochs required to terminate. It also was more likely to get caught on local minimums. This can be seen when analysing the output of the gene dataset, Sec. 12.1.4. There is often a rapid declination and then the error starts increasing again. Both of these are most likely due to the assumptions made when deriving the Quickprop algorithm. The ADAGRAD algorithm had a similar performance to the backward propagation algorithm but had less parameters to tune. The learning rate was set to a constant number for the duration of the algorithm, as opposed to the exponentially decaying learning rate used in forward propagation. This makes it easier to apply this algorithm on many datasets, without the need to fine tune various parameters. Adding momentum to the backward propagation does not effect performance much for small datasets like proben1. It does however make it more likely to reach a global minimum. This can be seen when comparing the value and range of the test errors produced by this algorithm to that of backward propagation. Especially when there is less redundancy in the parameter space, such as in Sec. 12.1.2, one can see that backward propagation with momentum tends to achieve a lower error and a smaller test error range. Finally, it is important to note that the test errors achieved in most of these test are in-line with the results achieved in [15].

GA and PSO perform significantly worse than the gradient based algorithms. This is expected as the gradient based algorithms are able to exploit the geometry of the search space. These algorithms are less-likely to be trapped in local minima and hence often have a lower error rate when they converge. PSO is able to use artificial swam intelligence to gain better insight on the geometry of the search

space. Due to this, it is able to converge at a faster rate than the genetic algorithm. However, it is also more likely to get stuck on local minima than GA and hence it often plateaus at a higher error than GA. Although neural networks are able to perform well on a vast verity of problems, it is often better to use a algorithm which is more tailored to the specific problem. This can be seen when compiling the outputs of the Naive Bayesian classifier to that of the neural network. With the Bayesian classifier, we are able to achieve an accuracy of 89.97%, as opposed to the 70.19% achieved by the neural network. The Naive Bayesian classifier is far simpler to implement than the neural network and is able to achieve superior results. It is however possible to improve the neural network in order to achieve higher accuracy. An accuracy of 99.65% was achieved on the MNIST dataset using a deep neural network [18]. However, this would require a large amount or resources to implement and train.

## 14 Summary

Both gradient based methods and non-gradient based methods worked effectively when optimising the neural networks, however gradient based methods are able to converge a lot faster the GA and PSO. iRPROP+ showed excellent performance and was able to converge rapidly without any fine tuning. Most of the gradient based algorithms showed similar results on the proben1 dataset and achieved similar errors to the architectures tested in [15]. Quickprop was able to converge fast but suffered from stability issues caused by the assumptions made when deriving the algorithm. Finally, we note that a neural network, although powerful, may not be the right infrastructure for all machine learning problems. This can be seen with the MNIST dataset, where the Naive Bayesian classifier was able to out-perform the neural network with far less effort.

## 15 Conclusion

In this paper we were able to successfully implement a fully connected feed forward neural network. Various algorithms were used to train this network on both the proben1 and MNIST dataset. One of which, the iPROP+ algorithm, was able to achieve excellent results with minimal effort. It is also applicable to large neural networks with many hidden layers, as it is not subject to the vanishing gradient problem. Finally, we see that neural networks often require a fair amount of effort in order to perform better than other algorithms, which may be simpler in nature.

## 16 References

- [1] Y. LeCun and C. Cortes, “MNIST handwritten digit database,” 2010. [Online]. Available: <http://yann.lecun.com/exdb/mnist/>
- [2] K. Hornik, “Approximation capabilities of multilayer feedforward networks,” *Neural Networks*, vol. 4, no. 2, pp. 251 – 257, 1991. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/089360809190009T>
- [3] S. Hochreiter, “The vanishing gradient problem during learning recurrent neural nets and problem solutions,” *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, vol. 6, no. 02, pp. 107–116, 1998.
- [4] J. Martens, “Deep learning via hessian-free optimization.” in *ICML*, vol. 27, 2010, pp. 735–742.
- [5] C. Igel and M. Hüskens, “Improving the rprop learning algorithm,” in *Proceedings of the second international ICSC symposium on neural computation (NC 2000)*, vol. 2000. Citeseer, 2000, pp. 115–121.
- [6] M. Riedmiller and H. Braun, “Rprop-a fast adaptive learning algorithm,” in *Proc. of ISCIS VII), Universitat*. Citeseer, 1992.
- [7] ———, “A direct adaptive method for faster backpropagation learning: The rprop algorithm,” in *Neural Networks, 1993., IEEE International Conference on*. IEEE, 1993, pp. 586–591.
- [8] M. Riedmiller and I. Rprop, “Rprop-description and implementation details,” 1994.
- [9] S. E. Fahlman, “An empirical study of learning speed in back-propagation networks,” 1988.
- [10] J. Duchi, E. Hazan, and Y. Singer, “Adaptive subgradient methods for online learning and stochastic optimization,” *Journal of Machine Learning Research*, vol. 12, no. Jul, pp. 2121–2159, 2011.
- [11] V. Phansalkar and P. Sastry, “Analysis of the back-propagation algorithm with momentum,” *IEEE Transactions on Neural Networks*, vol. 5, no. 3, pp. 505–506, 1994.
- [12] C. Darwin, *On the origin of species*, 1859. Routledge, 2004.
- [13] J. Kennedy and R. Eberhart, “Particle swarm optimization,” in *Neural Networks, 1995. Proceedings., IEEE International Conference on*, vol. 4, Nov 1995, pp. 1942–1948 vol.4.
- [14] Y. Shi and R. Eberhart, “A modified particle swarm optimizer,” in *1998 IEEE International Conference on Evolutionary Computation Proceedings. IEEE World Congress on Computational Intelligence (Cat. No.98TH8360)*, May 1998, pp. 69–73.
- [15] L. Prechelt *et al.*, “Proben1: A set of neural network benchmark problems and benchmarking rules,” 1994.
- [16] L. Prechelt, “Early stopping-but when?” in *Neural Networks: Tricks of the trade*. Springer, 1998, pp. 55–69.
- [17] R. Poli, J. Kennedy, and T. Blackwell, “Particle swarm optimization,” *Swarm intelligence*, vol. 1, no. 1, pp. 33–57, 2007.
- [18] D. C. Cireşan, U. Meier, L. M. Gambardella, and J. Schmidhuber, “Deep, big, simple neural nets for handwritten digit recognition,” *Neural computation*, vol. 22, no. 12, pp. 3207–3220, 2010.

# 16 Appendix

## 16.1 cancer

This section displays the results obtained in the cancer dataset.

### 16.1.1 ADAGRAD

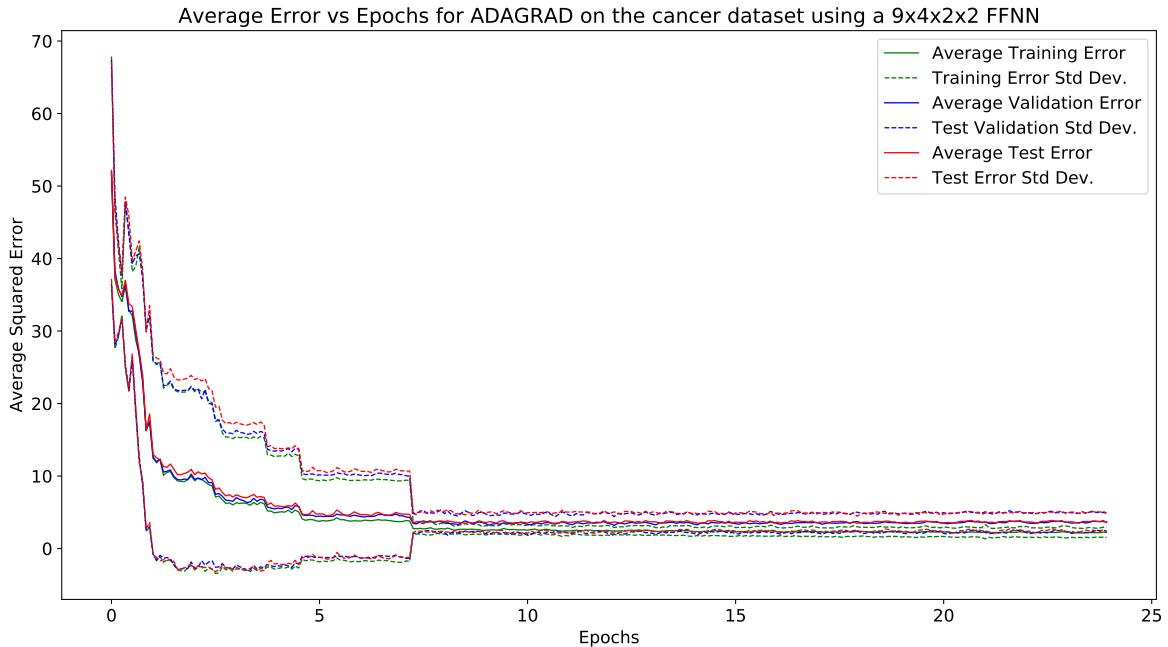
This section displays the results obtained using the ADAGRAD algorithm.

### 16.1.2 cancer

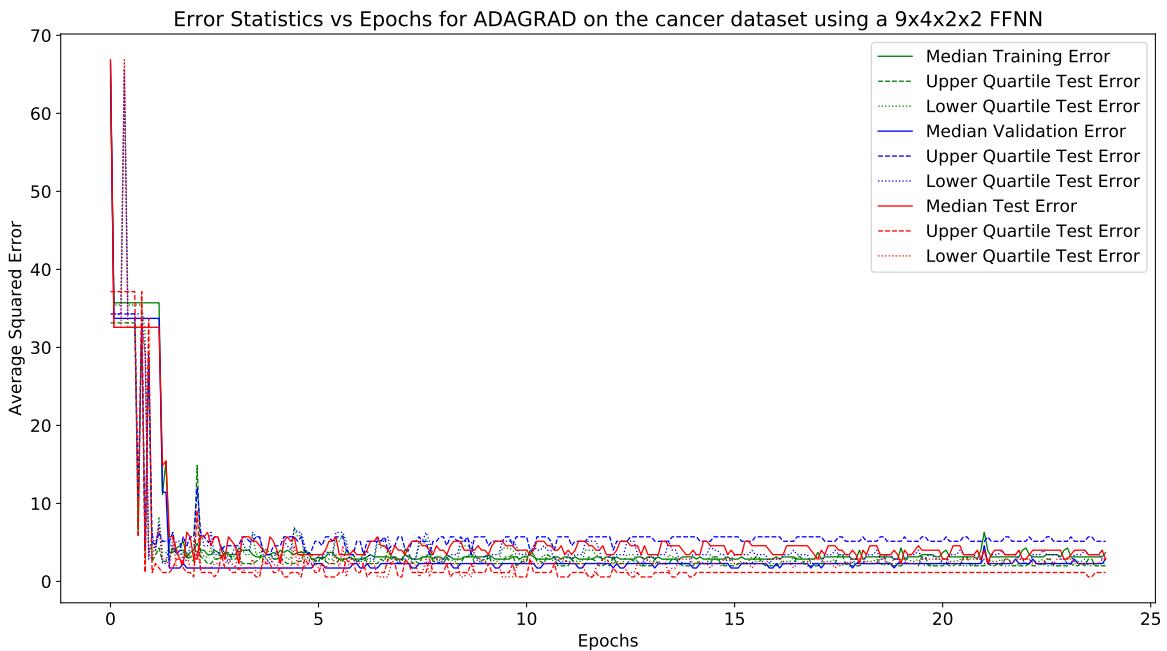
This section displays the results obtained using the cancer algorithm.

#### 9 × 4 × 2 × 2 Architecture:

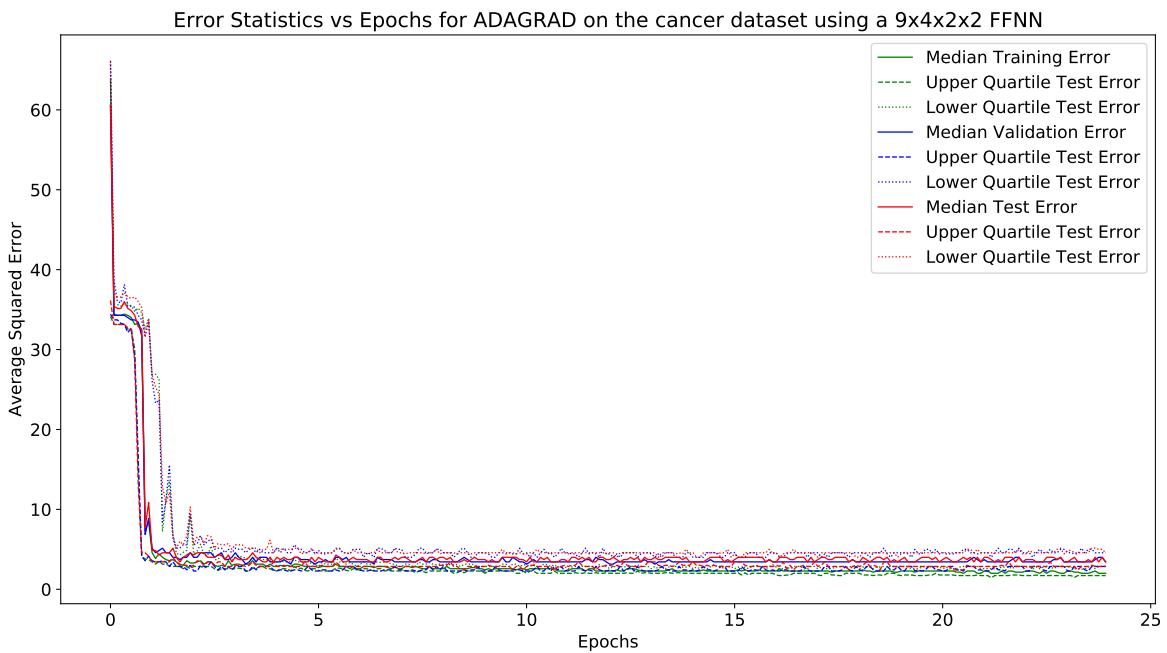
Fig. 77 shows the average and standard deviation of the test, training and validation errors. Fig. 78 shows the median and quartiles of the test, training and validation errors based on the best epoch. Finally, Fig. 79 shows the median and quartiles of the test, training and validation errors per mini-batch.



**Figure 77.** Graph of mean and standard deviation of errors vs epochs



**Figure 78.** Graph of test error vs epochs for the gradient based algorithms



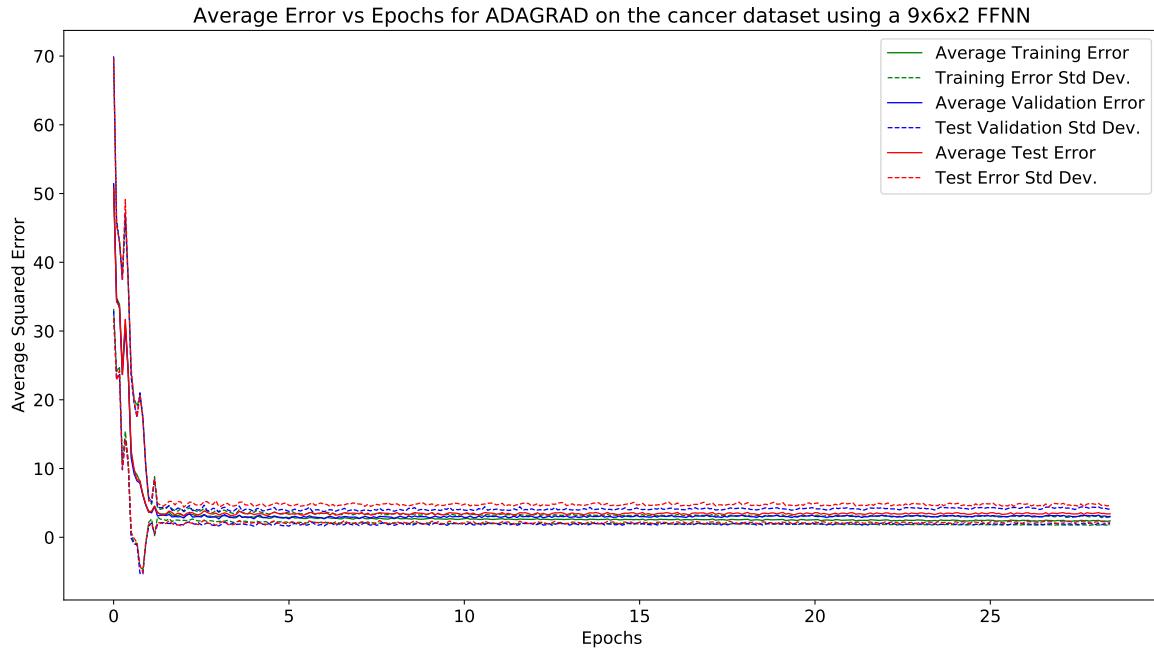
**Figure 79.** Graph of test error vs epochs for the gradient based algorithms

### 16.1.3 cancer

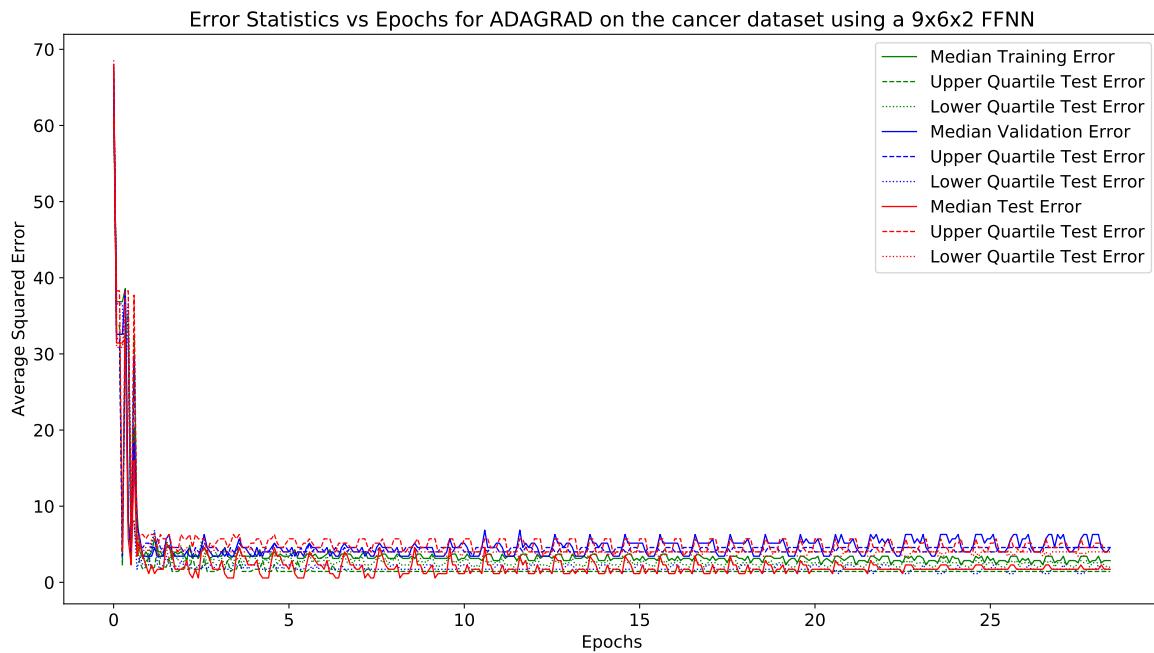
This section displays the results obtained using the cancer algorithm.

#### 9 × 6 × 2 Architecture:

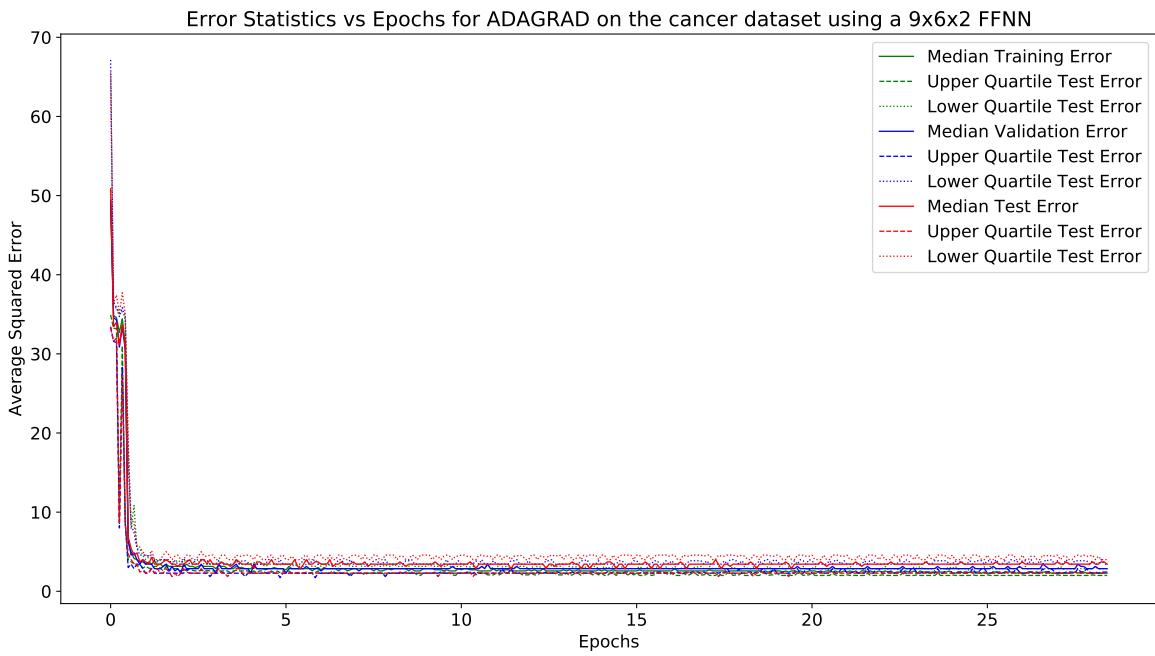
Fig. 80 shows the average and standard deviation of the test, training and validation errors. Fig. 81 shows the median and quartiles of the test, training and validation errors based on the best epoch. Finally, Fig. 82 shows the median and quartiles of the test, training and validation errors per mini-batch.



**Figure 80.** Graph of mean and standard deviation of errors vs epochs



**Figure 81.** Graph of test error vs epochs for the gradient based algorithms



**Figure 82.** Graph of test error vs epochs for the gradient based algorithms

#### 16.1.4 Backprop with Momentum

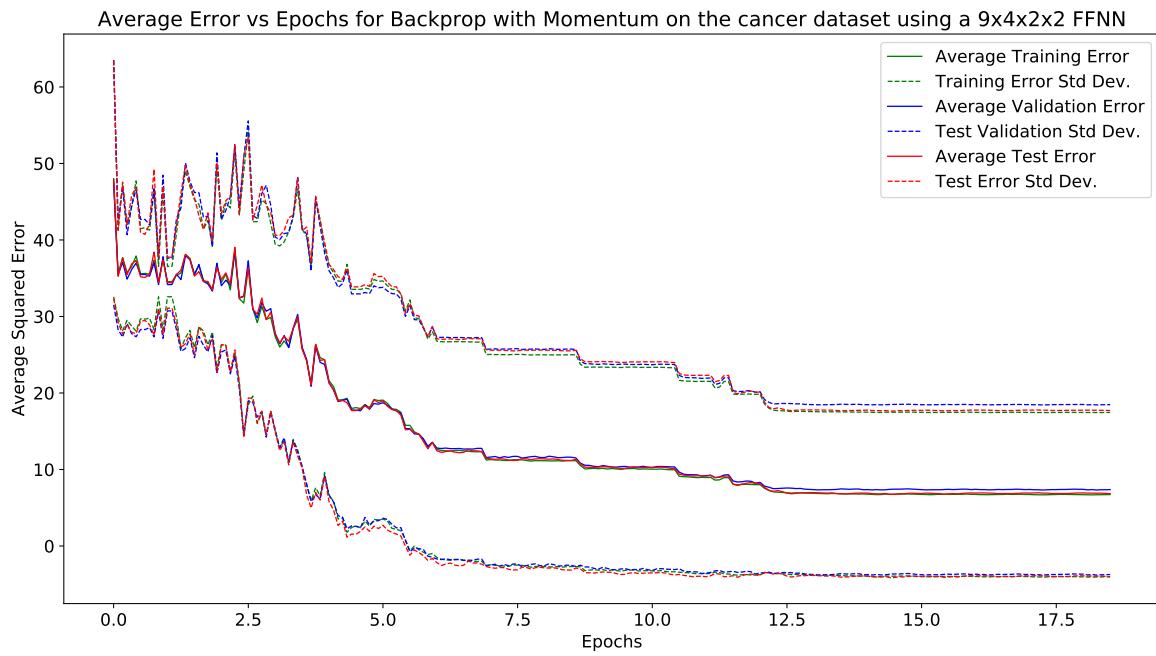
This section displays the results obtained using the Backprop with Momentum algorithm.

#### 16.1.5 cancer

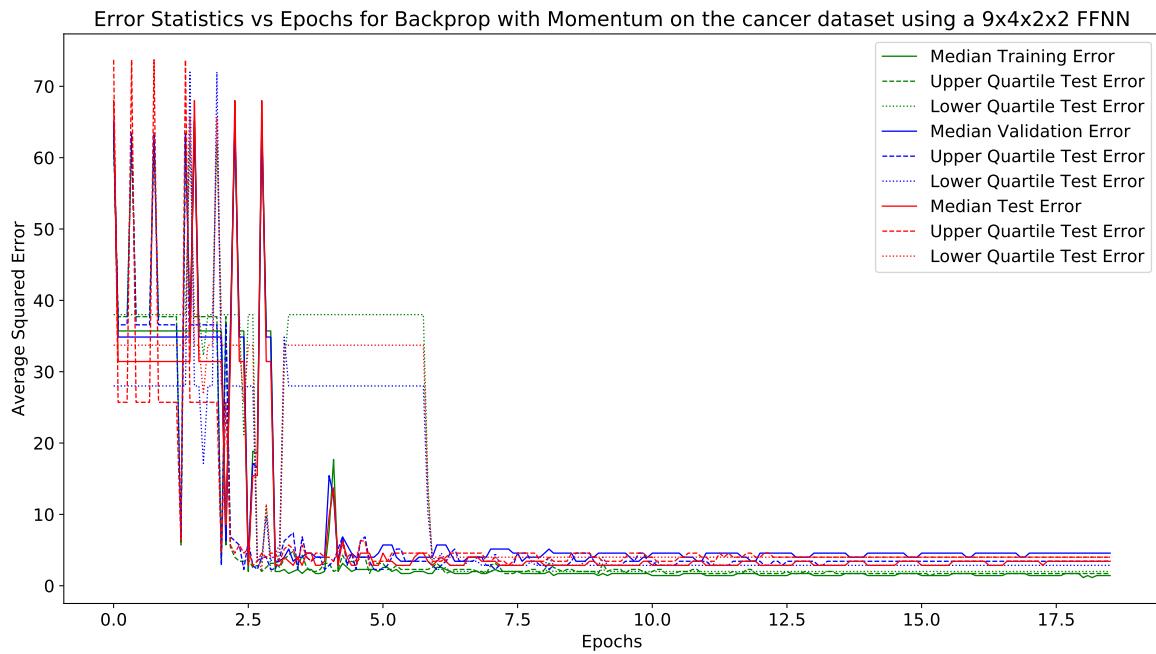
This section displays the results obtained using the cancer algorithm.

##### 9 × 4 × 2 × 2 Architecture:

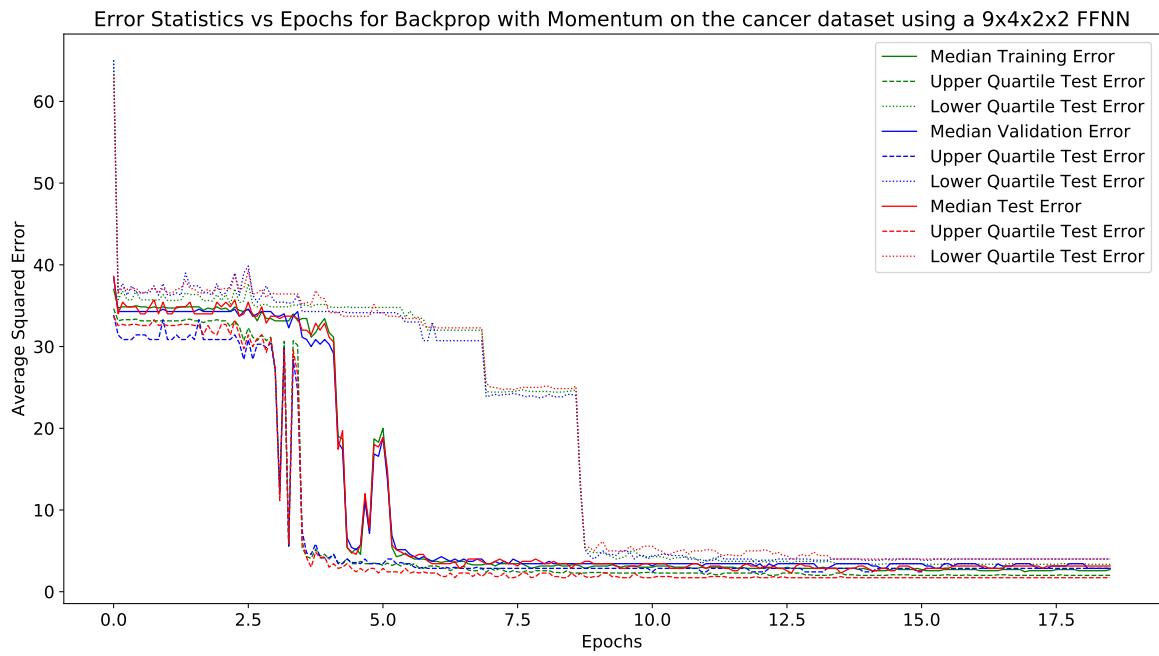
Fig. 83 shows the average and standard deviation of the test, training and validation errors. Fig. 84 shows the median and quartiles of the test, training and validation errors based on the best epoch. Finally, Fig. 85 shows the median and quartiles of the test, training and validation errors per mini-batch.



**Figure 83.** Graph of mean and standard deviation of errors vs epochs



**Figure 84.** Graph of test error vs epochs for the gradient based algorithms



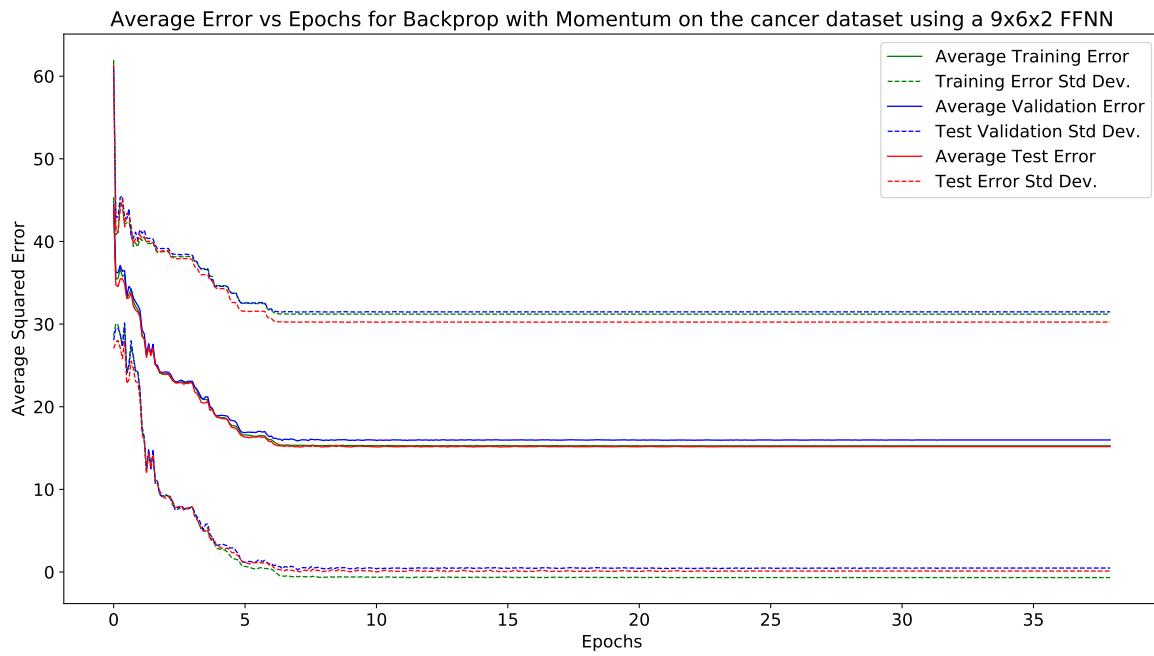
**Figure 85.** Graph of test error vs epochs for the gradient based algorithms

### 16.1.6 cancer

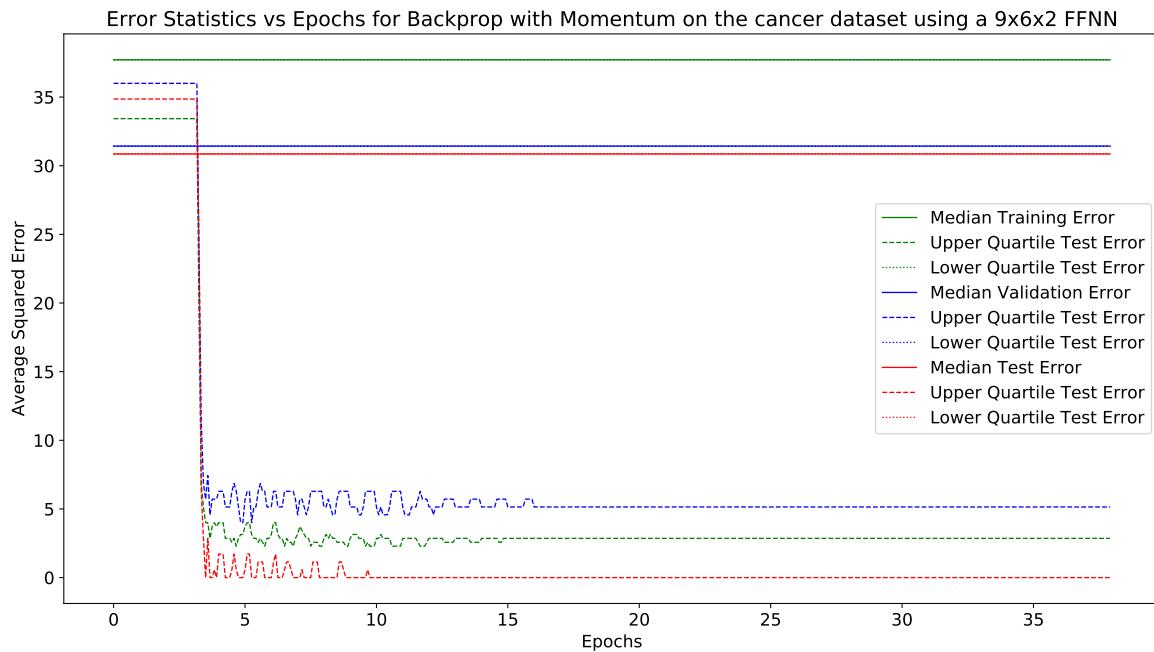
This section displays the results obtained using the cancer algorithm.

#### 9 × 6 × 2 Architecture:

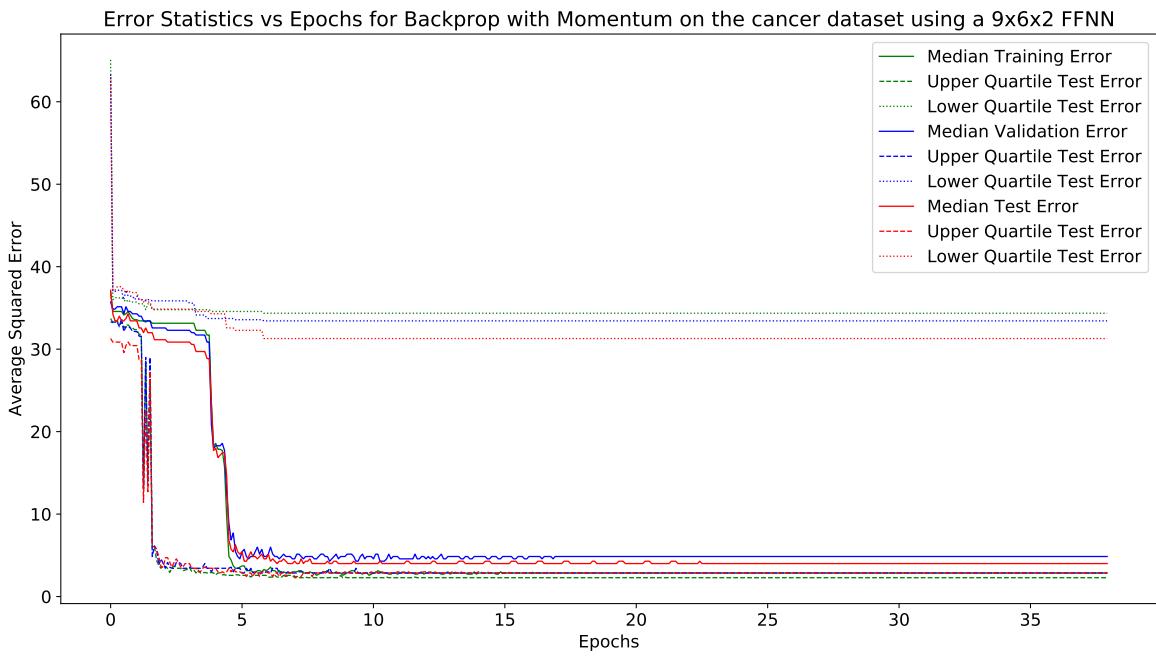
Fig. 86 shows the average and standard deviation of the test, training and validation errors. Fig. 87 shows the median and quartiles of the test, training and validation errors based on the best epoch. Finally, Fig. 88 shows the median and quartiles of the test, training and validation errors per mini-batch.



**Figure 86.** Graph of mean and standard deviation of errors vs epochs



**Figure 87.** Graph of test error vs epochs for the gradient based algorithms



**Figure 88.** Graph of test error vs epochs for the gradient based algorithms

### 16.1.7 back-propogation

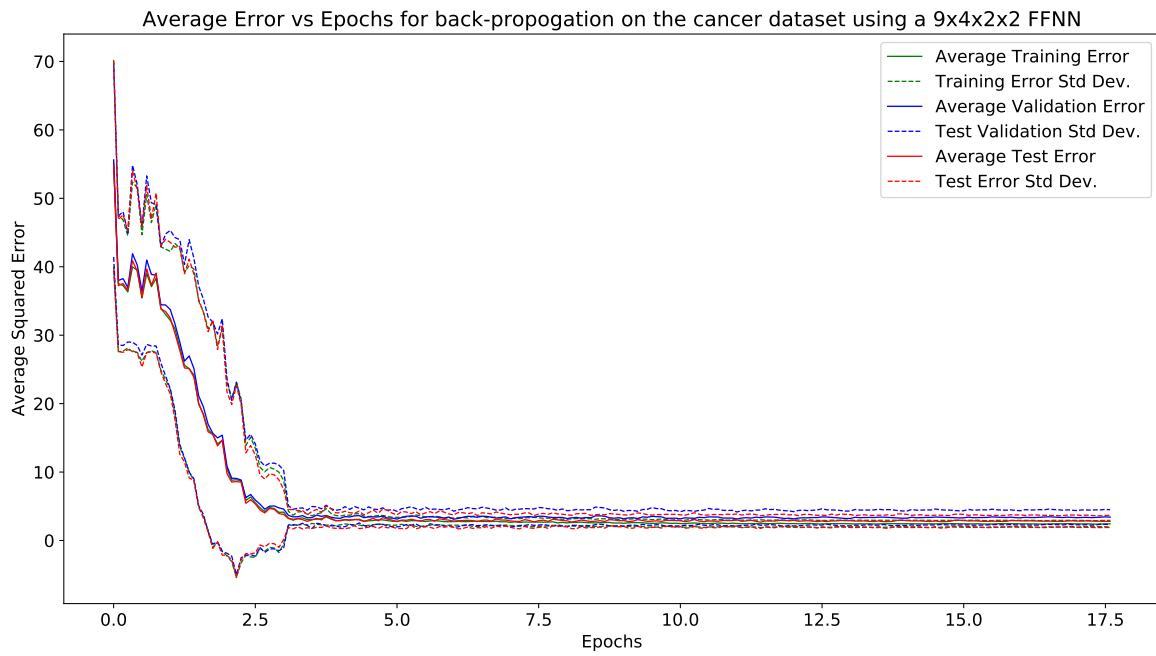
This section displays the results obtained using the back-propogation algorithm.

### 16.1.8 cancer

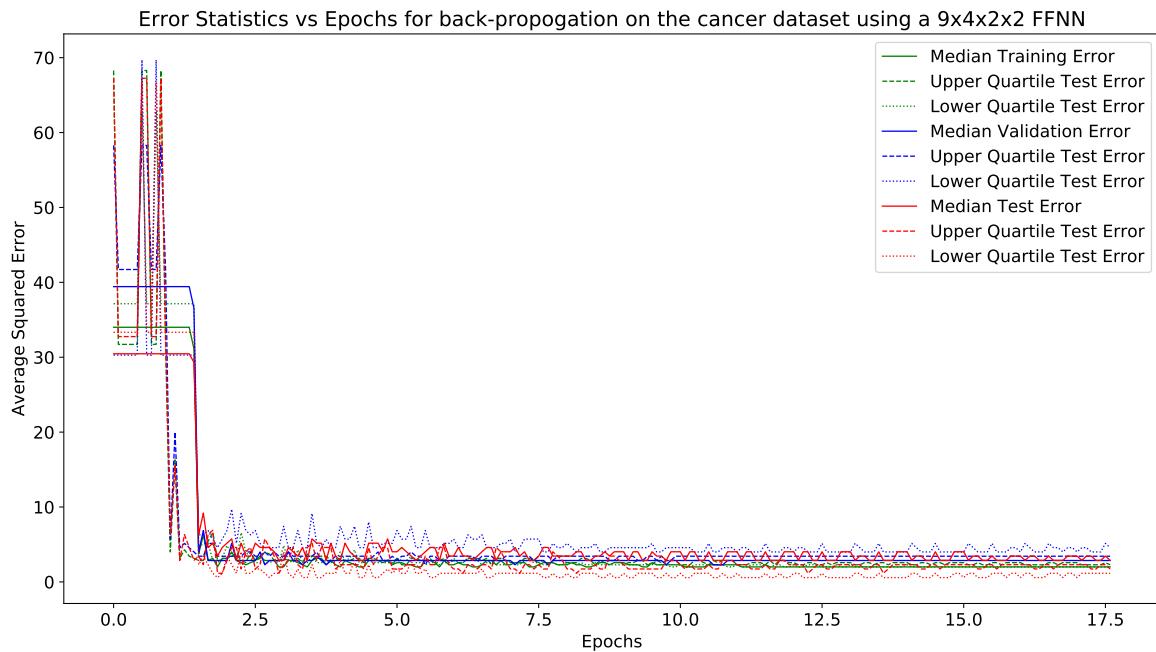
This section displays the results obtained using the cancer algorithm.

#### 9 × 4 × 2 × 2 Architecture:

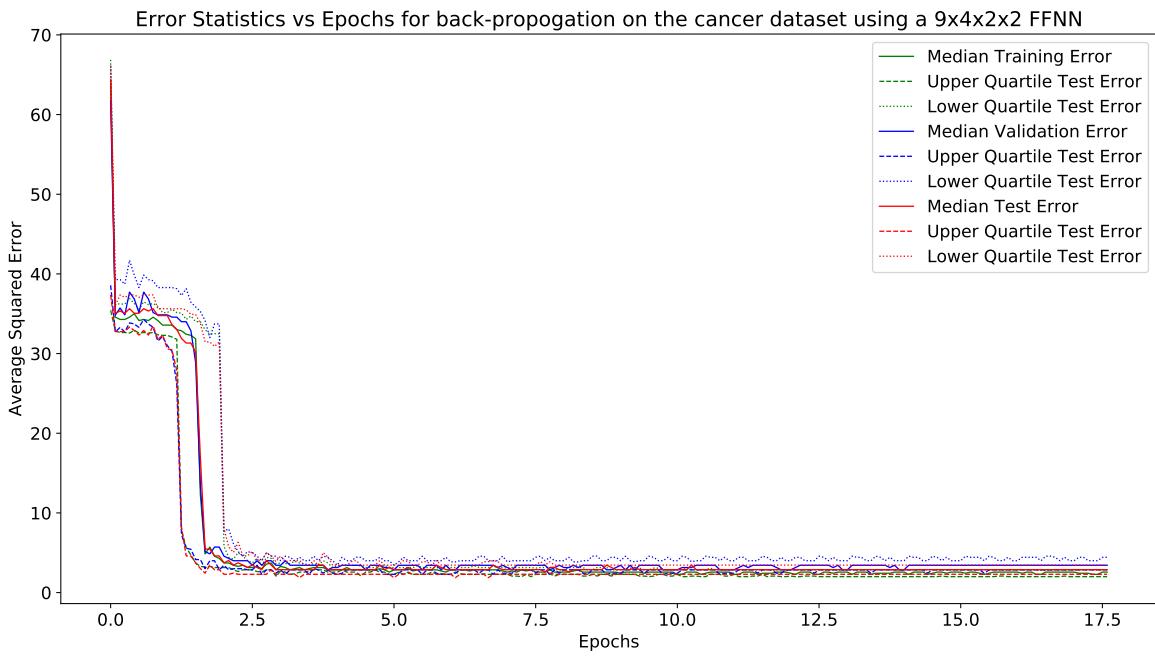
Fig. 89 shows the average and standard deviation of the test, training and validation errors. Fig. 90 shows the median and quartiles of the test, training and validation errors based on the best epoch. Finally, Fig. 91 shows the median and quartiles of the test, training and validation errors per mini-batch.



**Figure 89.** Graph of mean and standard deviation of errors vs epochs



**Figure 90.** Graph of test error vs epochs for the gradient based algorithms



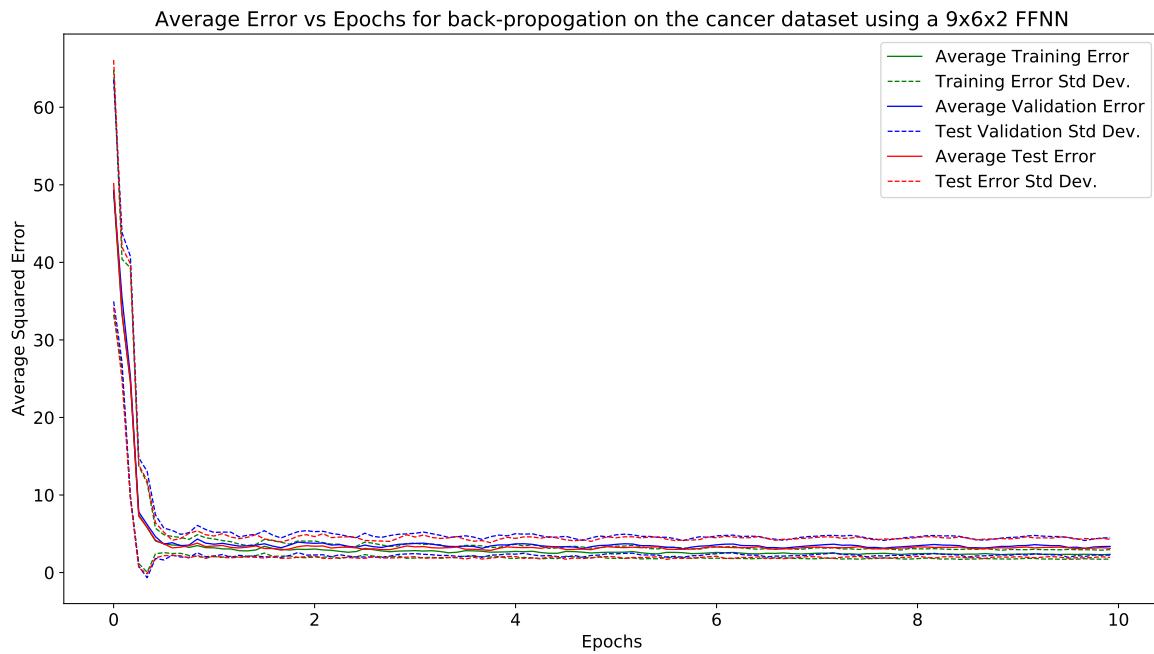
**Figure 91.** Graph of test error vs epochs for the gradient based algorithms

### 16.1.9 cancer

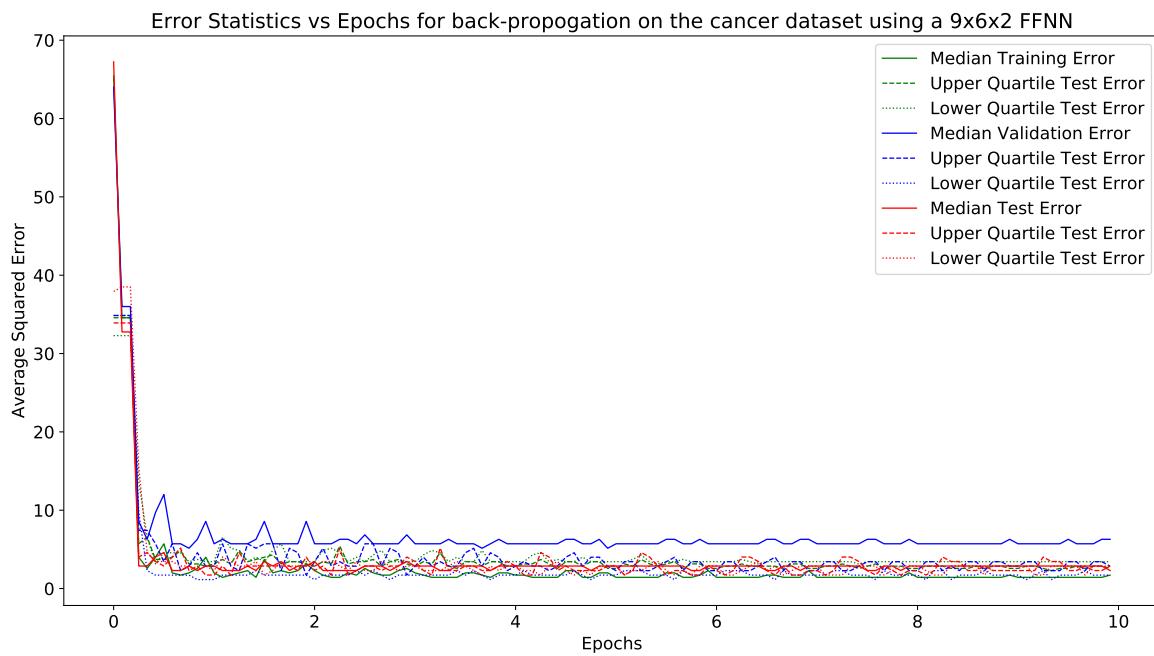
This section displays the results obtained using the cancer algorithm.

#### 9 × 6 × 2 Architecture:

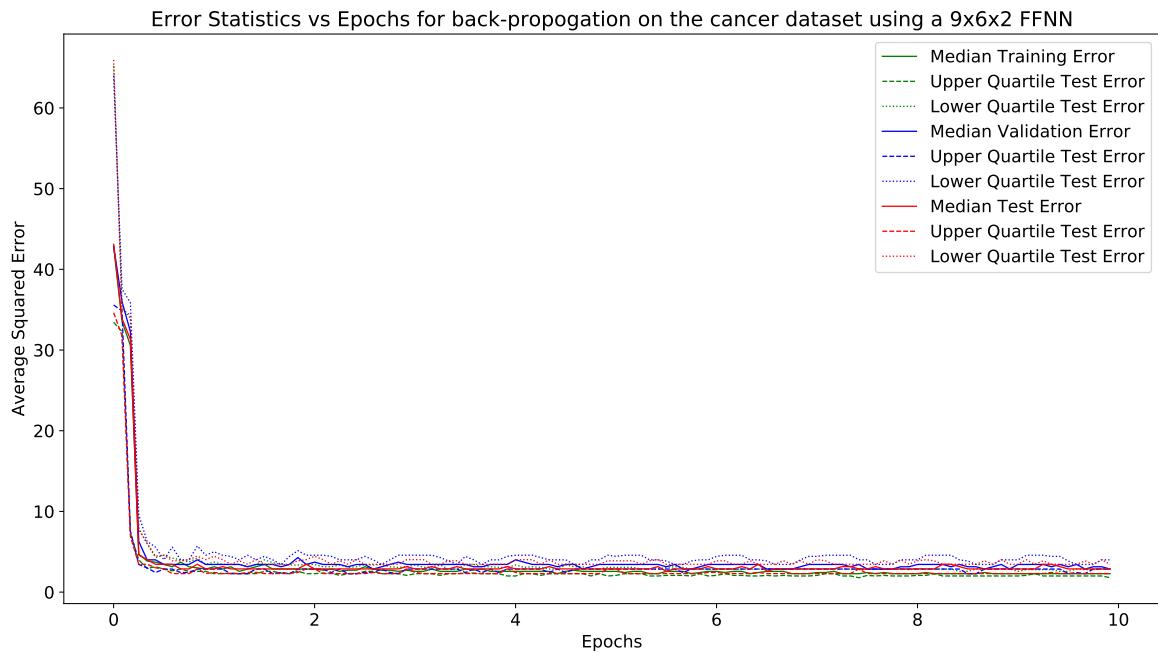
Fig. 92 shows the average and standard deviation of the test, training and validation errors. Fig. 93 shows the median and quartiles of the test, training and validation errors based on the best epoch. Finally, Fig. 94 shows the median and quartiles of the test, training and validation errors per mini-batch.



**Figure 92.** Graph of mean and standard deviation of errors vs epochs



**Figure 93.** Graph of test error vs epochs for the gradient based algorithms



**Figure 94.** Graph of test error vs epochs for the gradient based algorithms

### 16.1.10 GA1

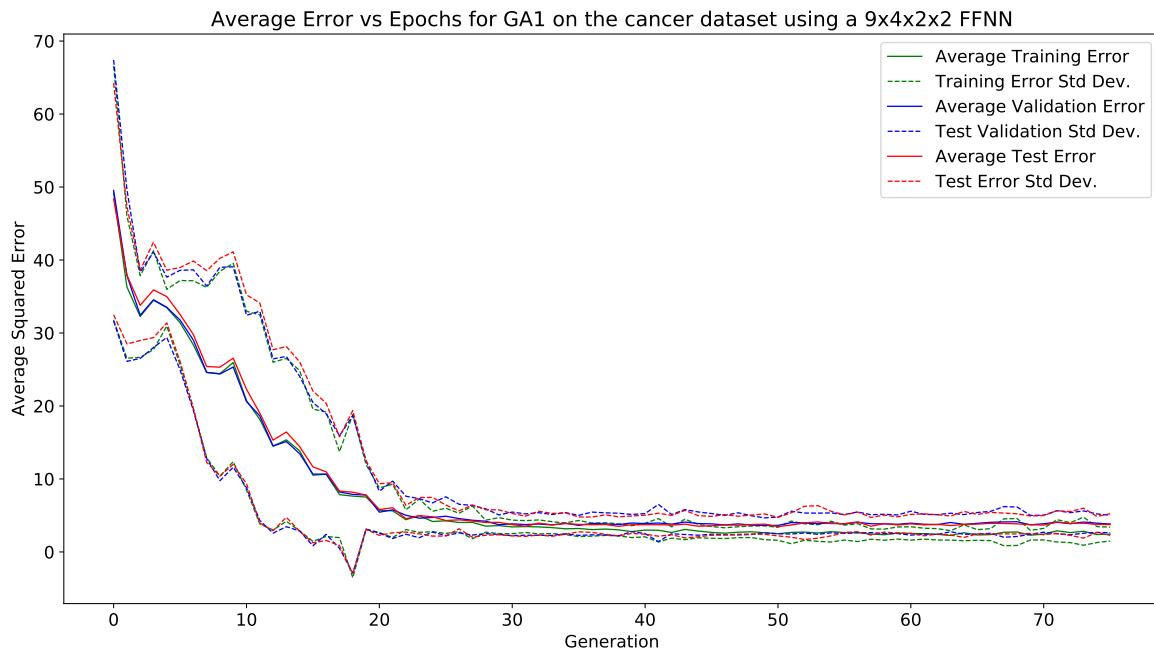
This section displays the results obtained using the GA1 algorithm.

### 16.1.11 cancer

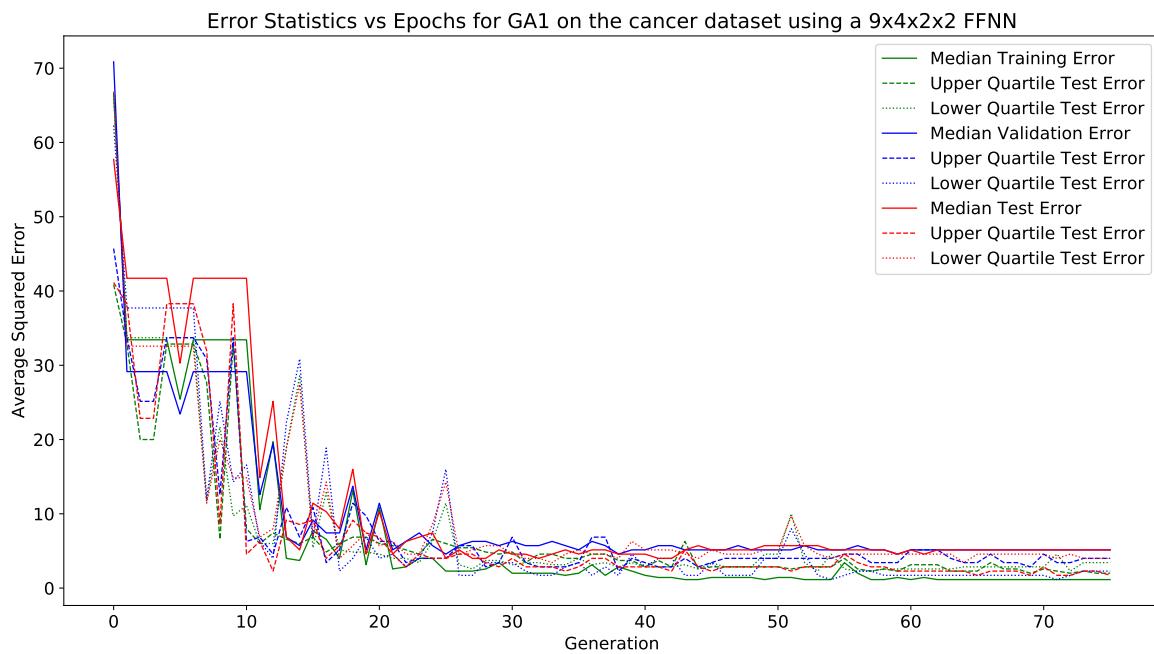
This section displays the results obtained using the cancer algorithm.

#### 9 × 4 × 2 × 2 Architecture:

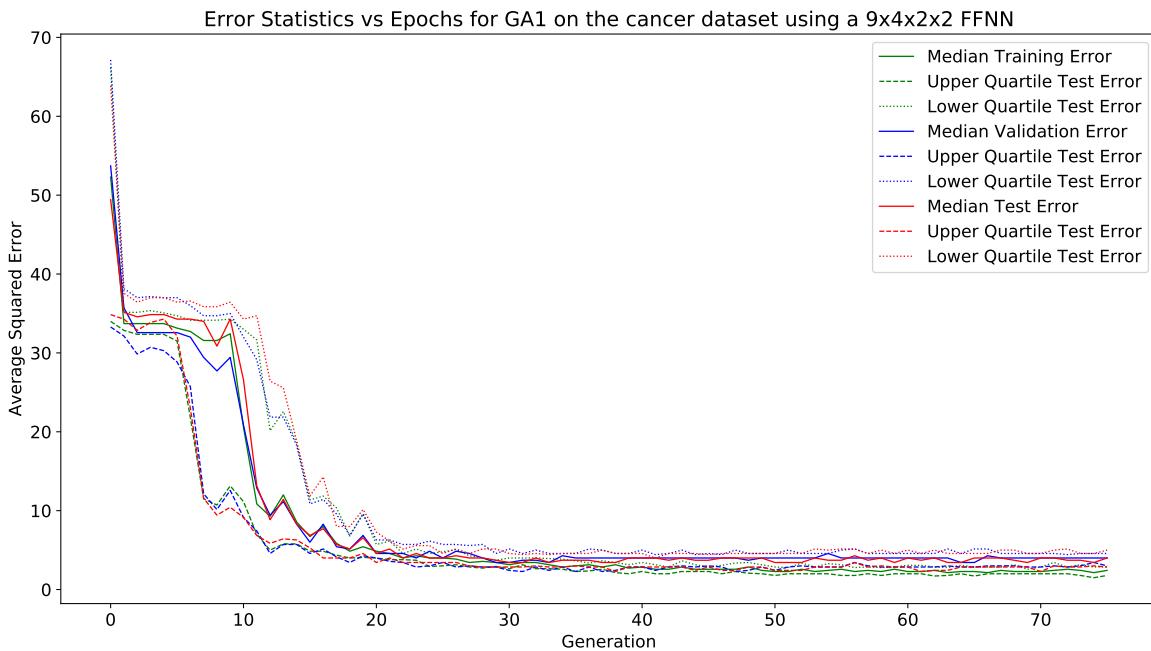
Fig. 95 shows the average and standard deviation of the test, training and validation errors. Fig. 96 shows the median and quartiles of the test, training and validation errors based on the best epoch. Finally, Fig. 97 shows the median and quartiles of the test, training and validation errors per mini-batch.



**Figure 95.** Graph of mean and standard deviation of errors vs epochs



**Figure 96.** Graph of test error vs epochs for the gradient based algorithms



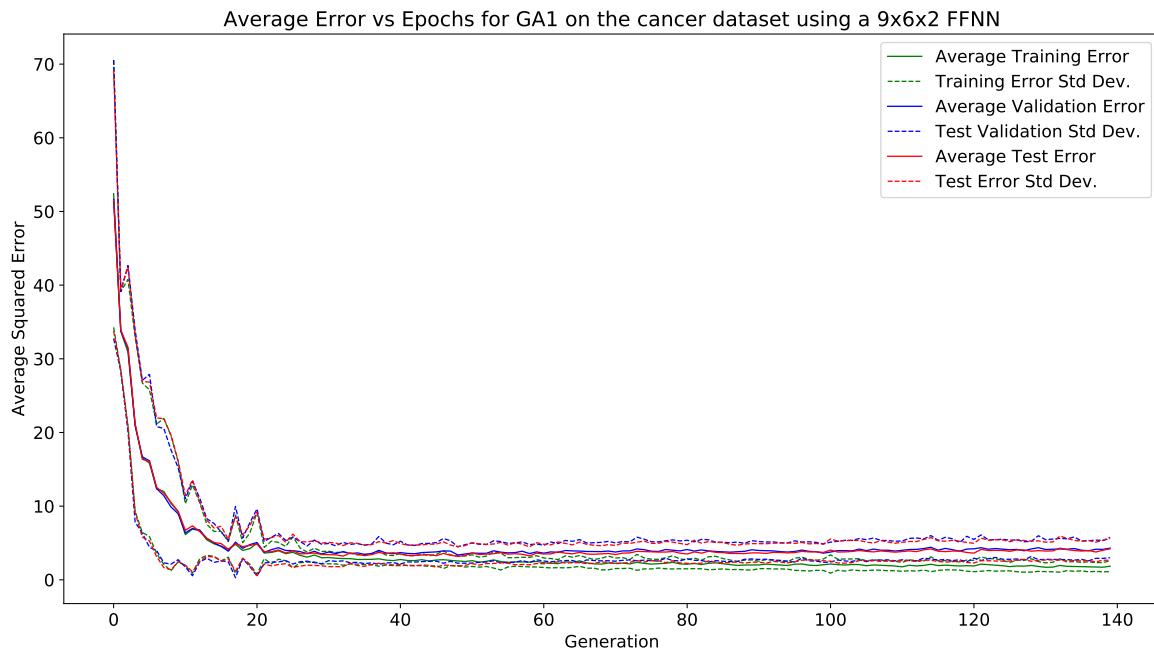
**Figure 97.** Graph of test error vs epochs for the gradient based algorithms

### 16.1.12 cancer

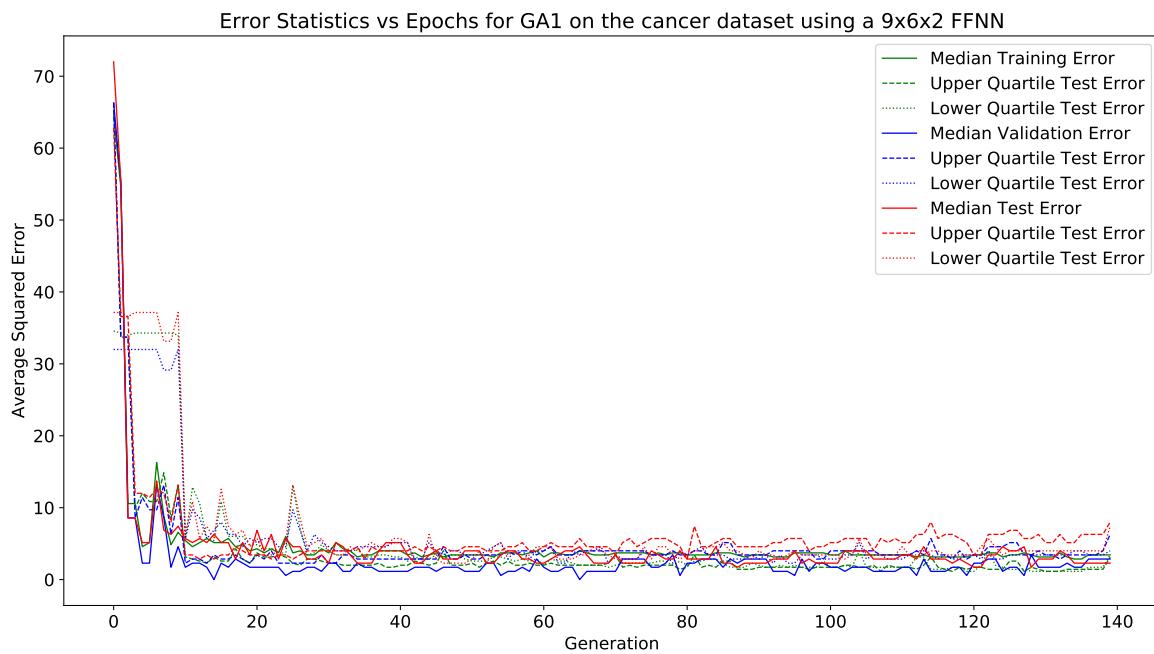
This section displays the results obtained using the cancer algorithm.

#### 9 × 6 × 2 Architecture:

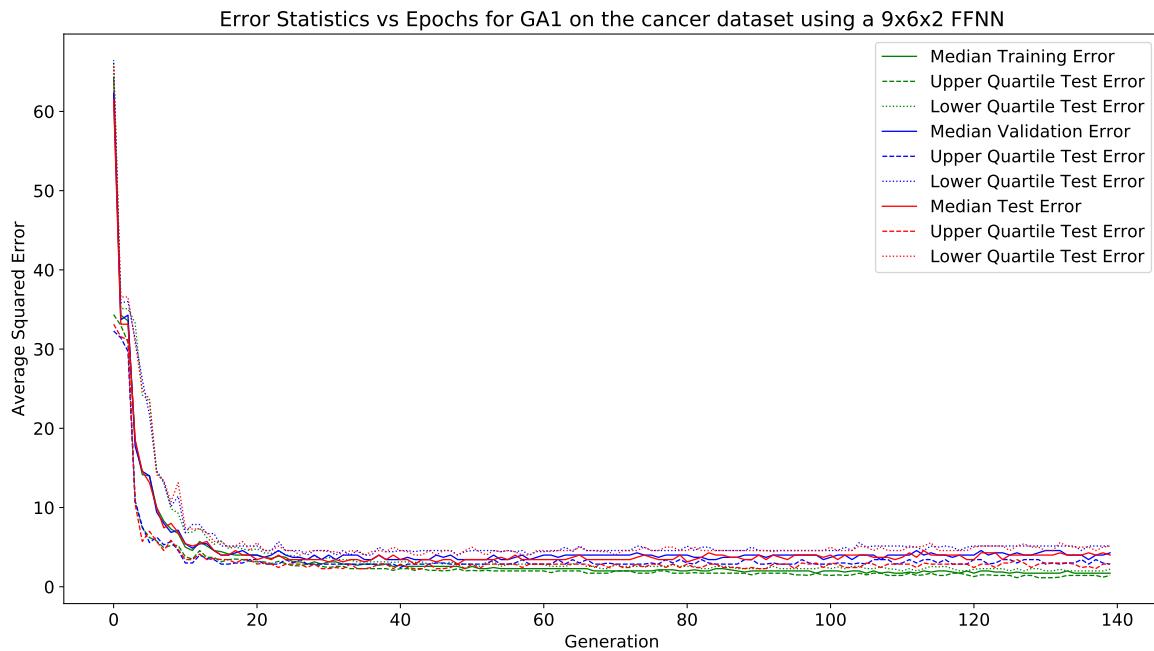
Fig. 98 shows the average and standard deviation of the test, training and validation errors. Fig. 99 shows the median and quartiles of the test, training and validation errors based on the best epoch. Finally, Fig. 100 shows the median and quartiles of the test, training and validation errors per mini-batch.



**Figure 98.** Graph of mean and standard deviation of errors vs epochs



**Figure 99.** Graph of test error vs epochs for the gradient based algorithms



**Figure 100.** Graph of test error vs epochs for the gradient based algorithms

### 16.1.13 GA2

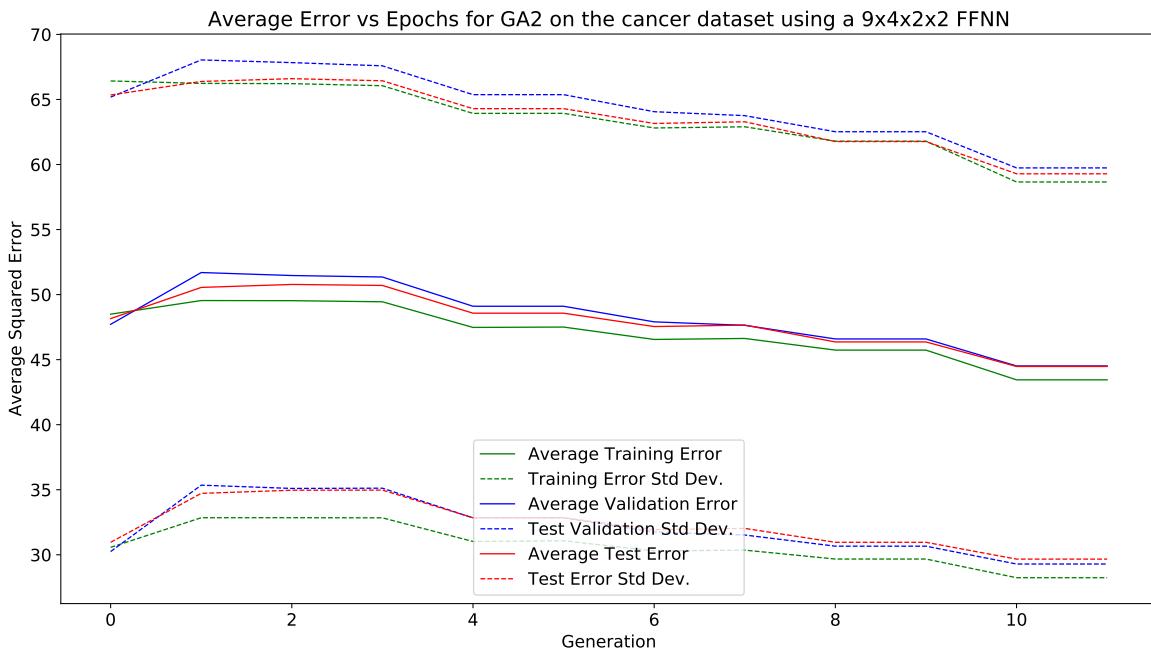
This section displays the results obtained using the GA2 algorithm.

### 16.1.14 cancer

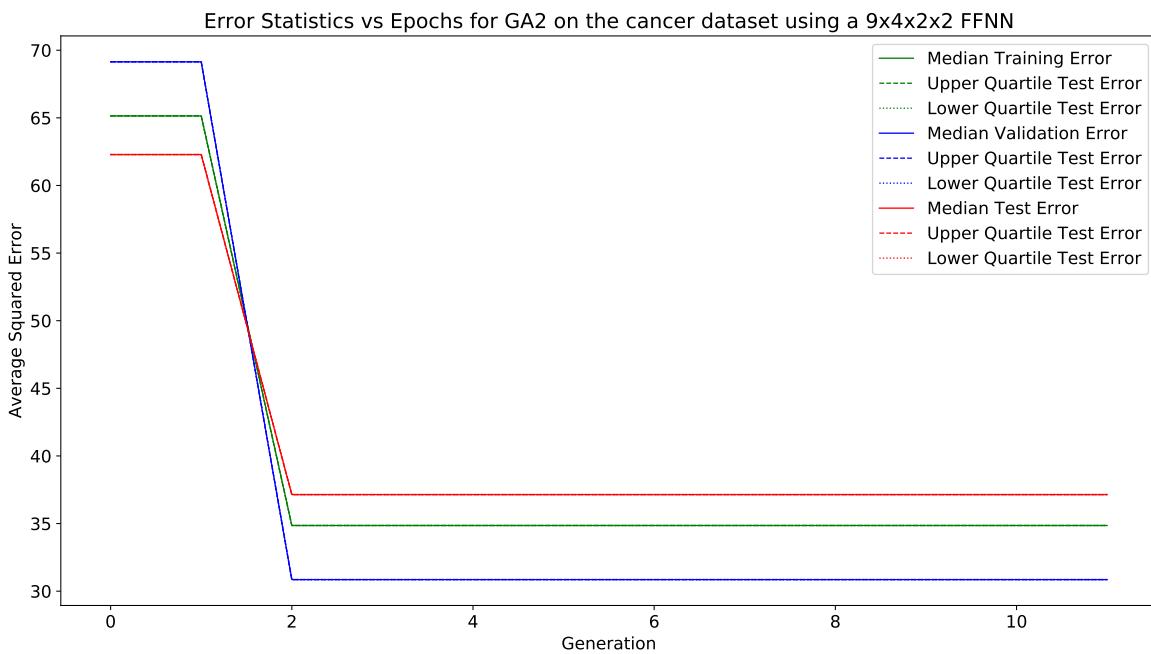
This section displays the results obtained using the cancer algorithm.

#### 9 × 4 × 2 × 2 Architecture:

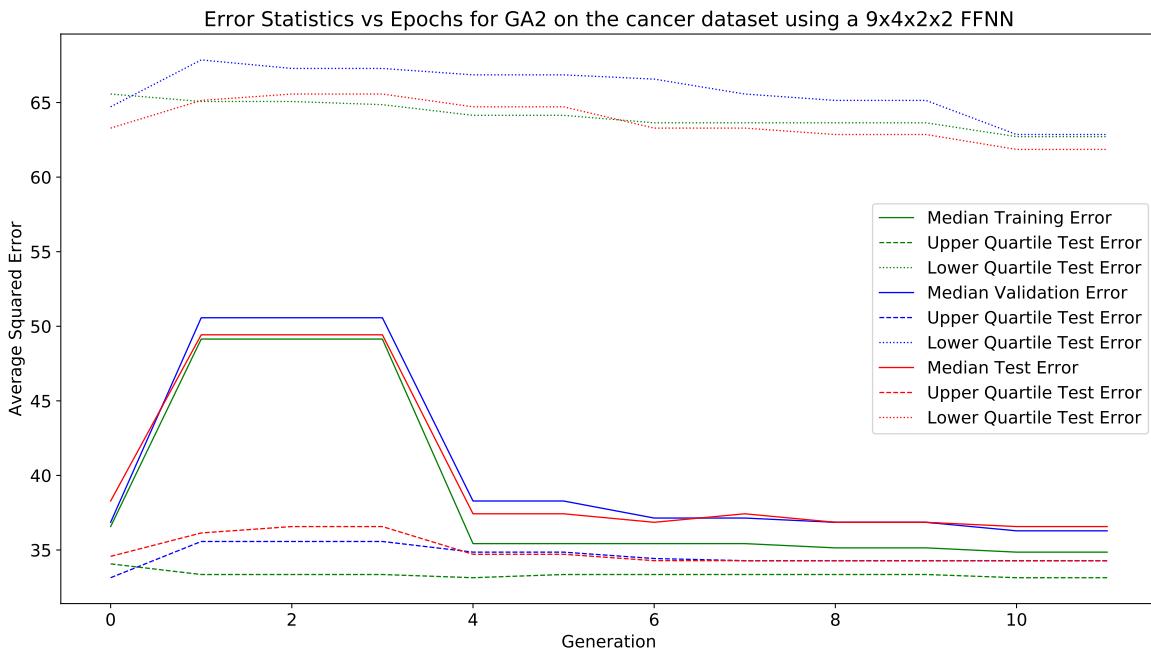
Fig. 101 shows the average and standard deviation of the test, training and validation errors. Fig. 102 shows the median and quartiles of the test, training and validation errors based on the best epoch. Finally, Fig. 103 shows the median and quartiles of the test, training and validation errors per mini-batch.



**Figure 101.** Graph of mean and standard deviation of errors vs epochs



**Figure 102.** Graph of test error vs epochs for the gradient based algorithms



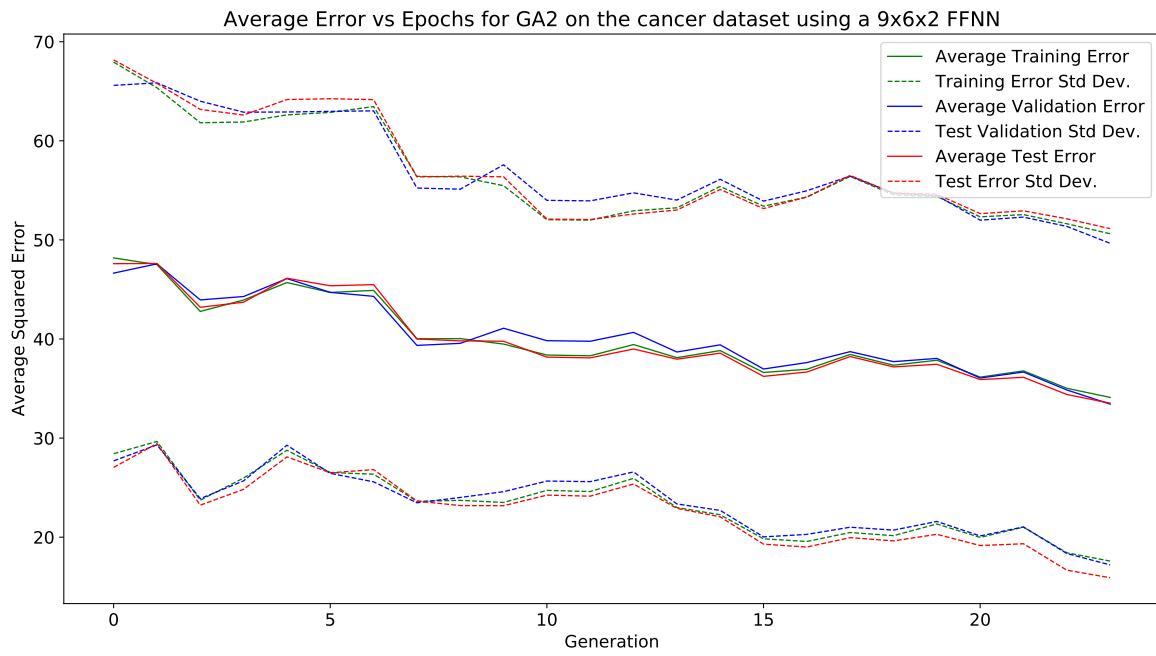
**Figure 103.** Graph of test error vs epochs for the gradient based algorithms

### 16.1.15 cancer

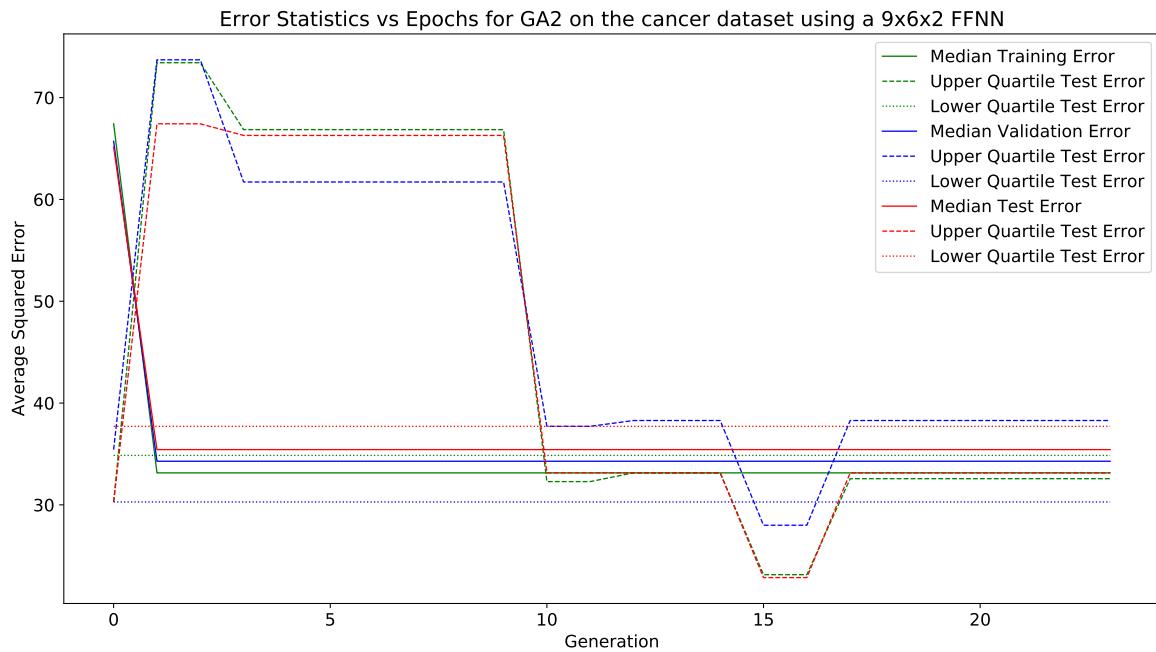
This section displays the results obtained using the cancer algorithm.

#### 9 × 6 × 2 Architecture:

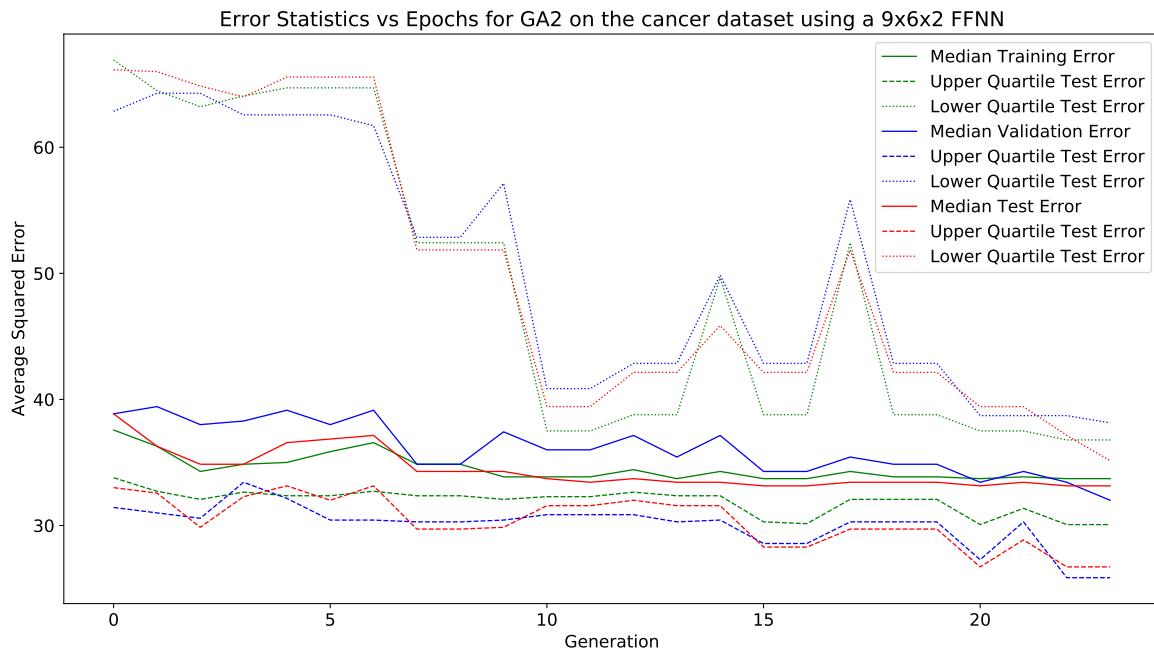
Fig. 104 shows the average and standard deviation of the test, training and validation errors. Fig. 105 shows the median and quartiles of the test, training and validation errors based on the best epoch. Finally, Fig. 106 shows the median and quartiles of the test, training and validation errors per mini-batch.



**Figure 104.** Graph of mean and standard deviation of errors vs epochs



**Figure 105.** Graph of test error vs epochs for the gradient based algorithms



**Figure 106.** Graph of test error vs epochs for the gradient based algorithms

### 16.1.16 GA3

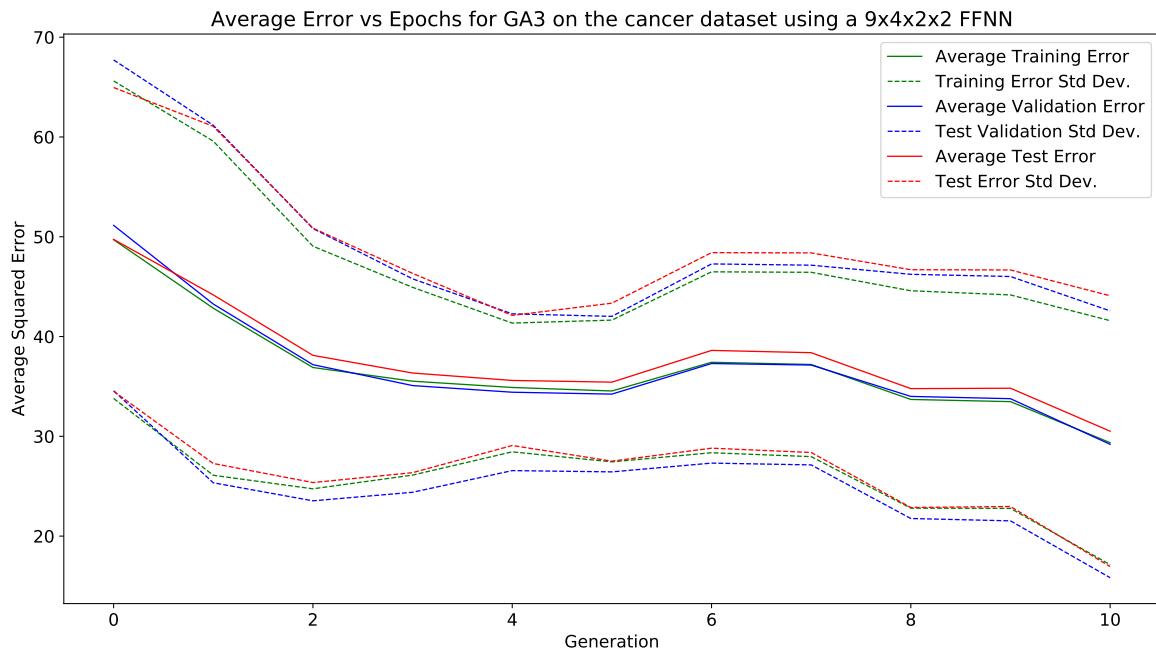
This section displays the results obtained using the GA3 algorithm.

### 16.1.17 cancer

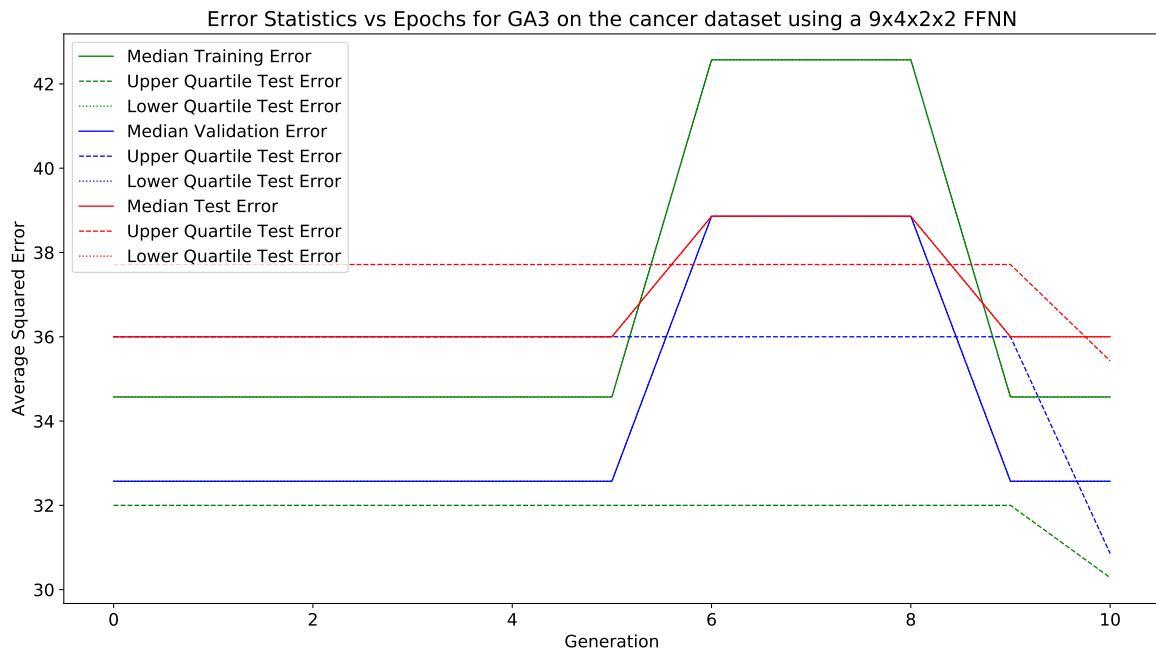
This section displays the results obtained using the cancer algorithm.

#### 9 × 4 × 2 × 2 Architecture:

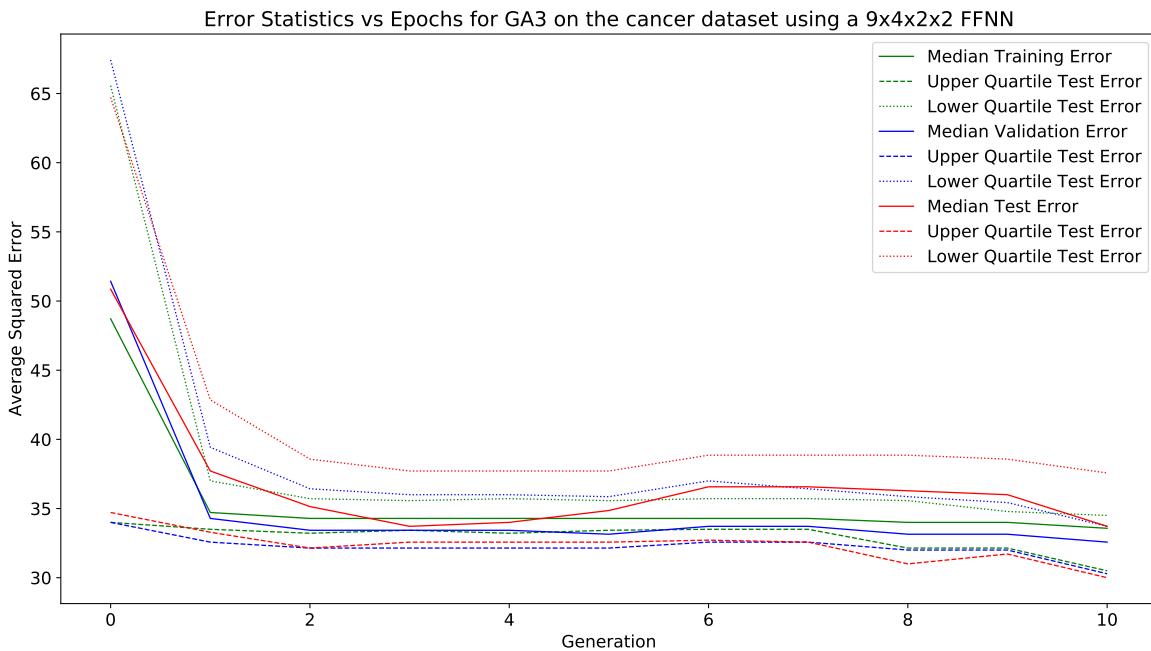
Fig. 107 shows the average and standard deviation of the test, training and validation errors. Fig. 108 shows the median and quartiles of the test, training and validation errors based on the best epoch. Finally, Fig. 109 shows the median and quartiles of the test, training and validation errors per mini-batch.



**Figure 107.** Graph of mean and standard deviation of errors vs epochs



**Figure 108.** Graph of test error vs epochs for the gradient based algorithms



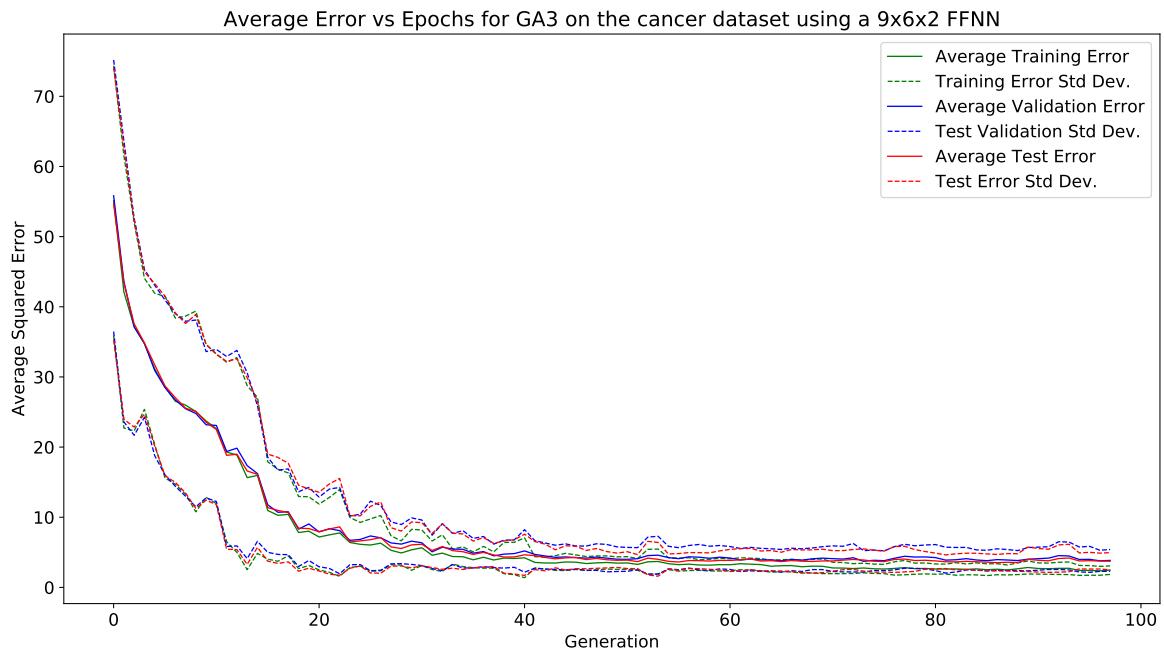
**Figure 109.** Graph of test error vs epochs for the gradient based algorithms

### 16.1.18 cancer

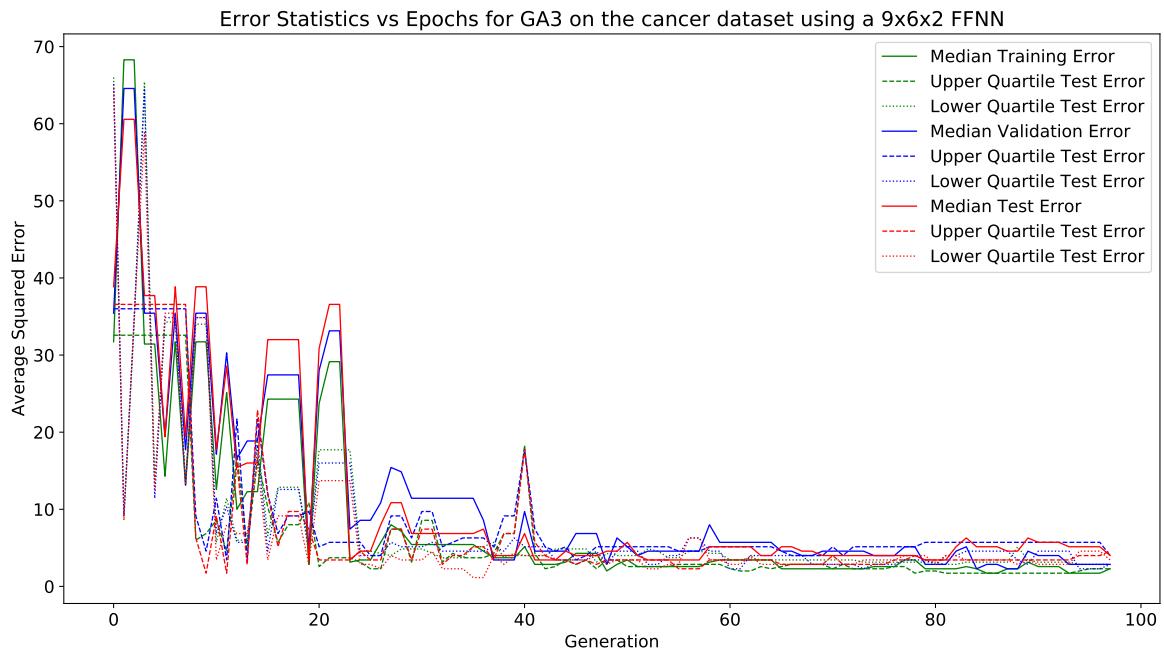
This section displays the results obtained using the cancer algorithm.

#### 9 × 6 × 2 Architecture:

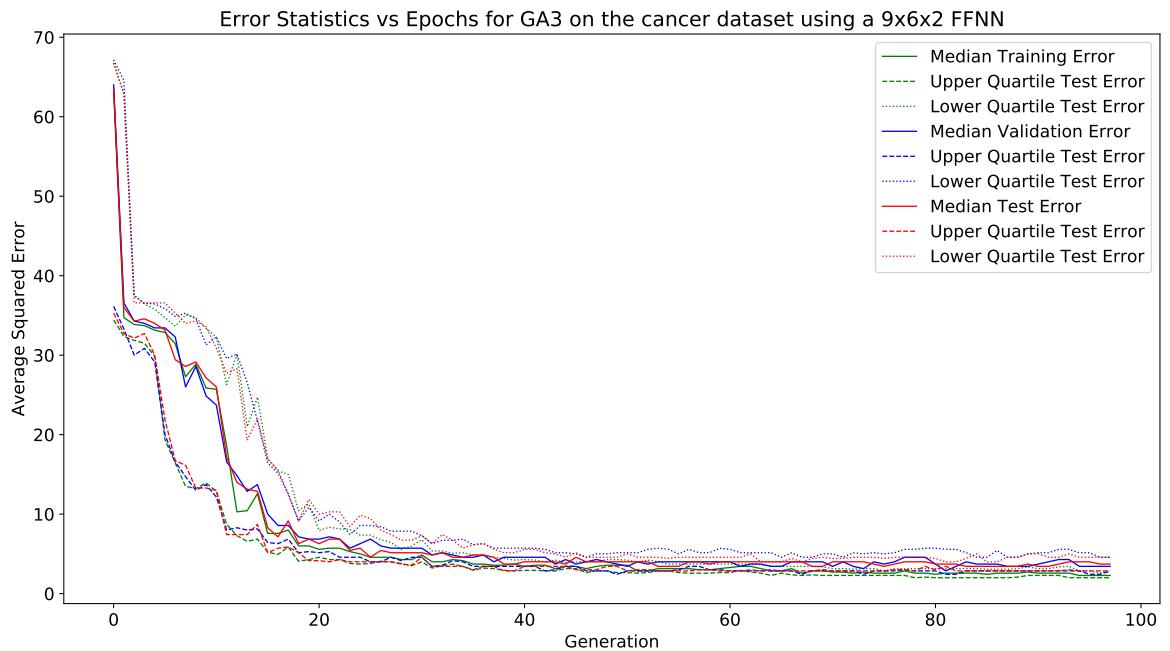
Fig. 110 shows the average and standard deviation of the test, training and validation errors. Fig. 111 shows the median and quartiles of the test, training and validation errors based on the best epoch. Finally, Fig. 112 shows the median and quartiles of the test, training and validation errors per mini-batch.



**Figure 110.** Graph of mean and standard deviation of errors vs epochs



**Figure 111.** Graph of test error vs epochs for the gradient based algorithms



**Figure 112.** Graph of test error vs epochs for the gradient based algorithms

### 16.1.19 iRPROP-

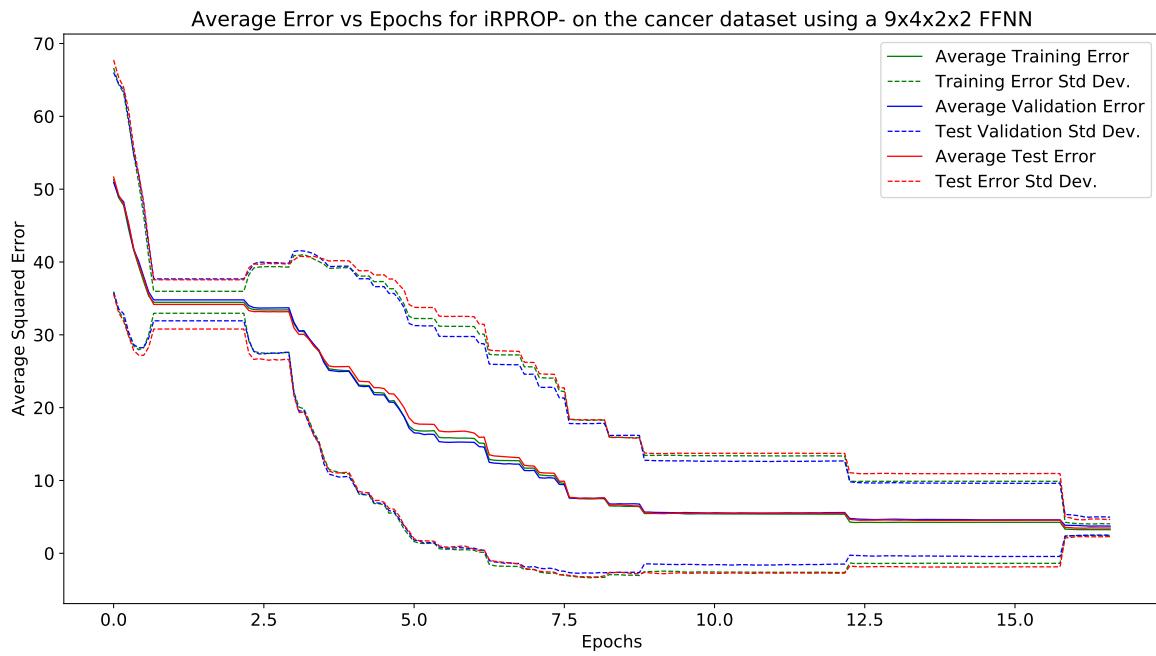
This section displays the results obtained using the iRPROP- algorithm.

### 16.1.20 cancer

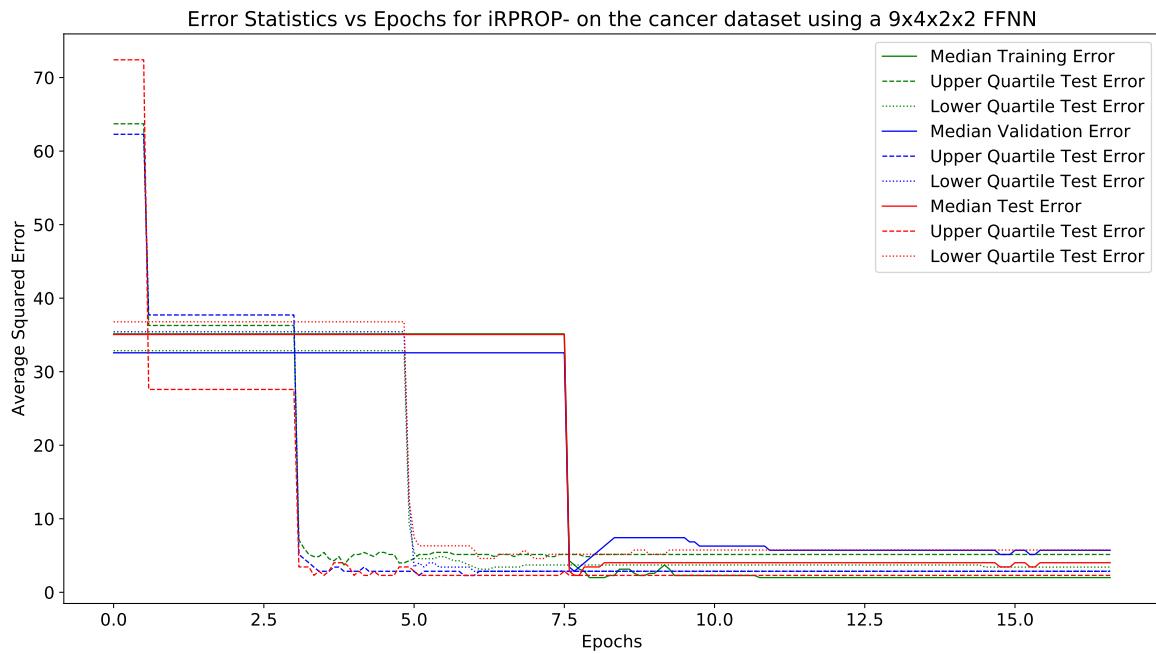
This section displays the results obtained using the cancer algorithm.

#### 9 × 4 × 2 × 2 Architecture:

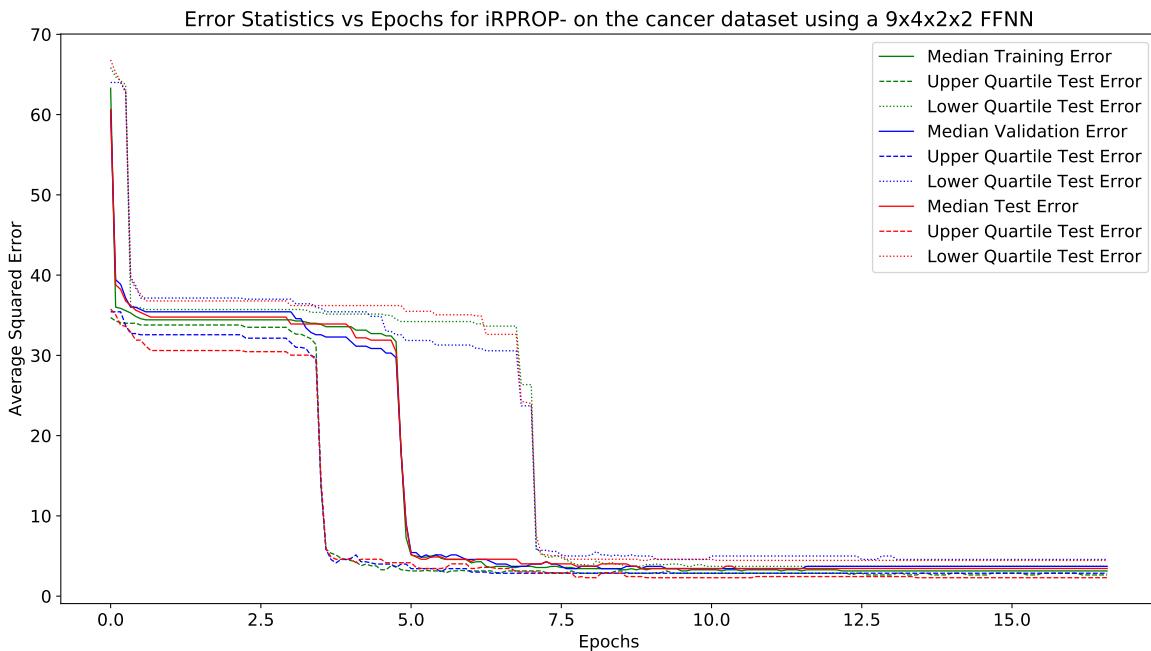
Fig. 113 shows the average and standard deviation of the test, training and validation errors. Fig. 114 shows the median and quartiles of the test, training and validation errors based on the best epoch. Finally, Fig. 115 shows the median and quartiles of the test, training and validation errors per mini-batch.



**Figure 113.** Graph of mean and standard deviation of errors vs epochs



**Figure 114.** Graph of test error vs epochs for the gradient based algorithms



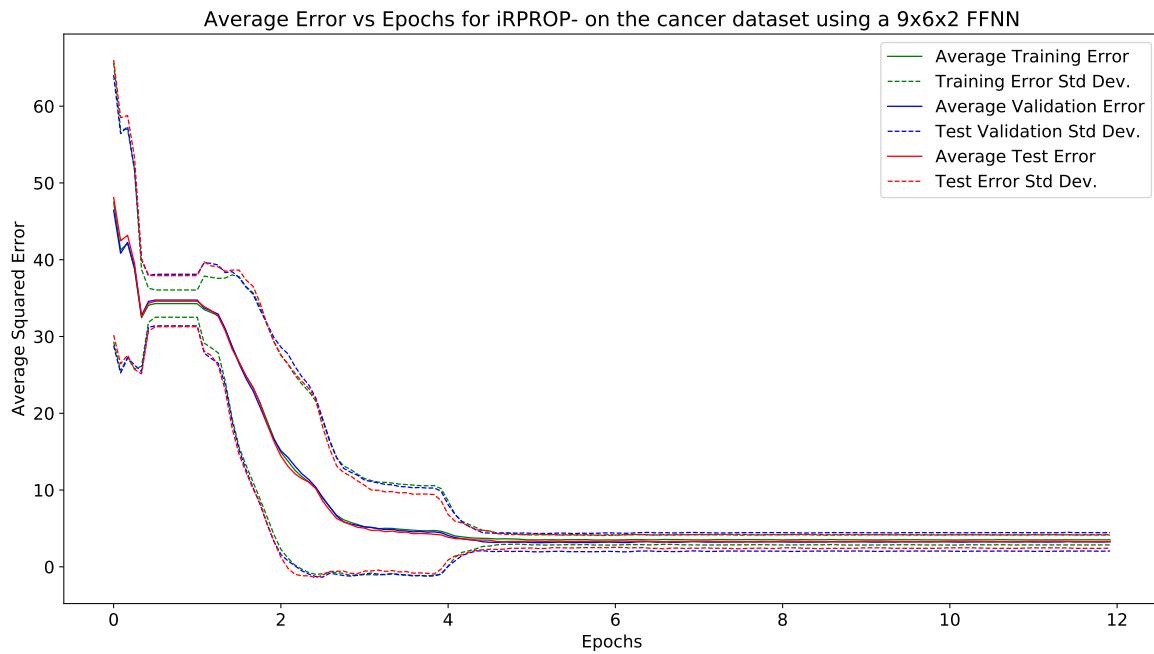
**Figure 115.** Graph of test error vs epochs for the gradient based algorithms

### 16.1.21 cancer

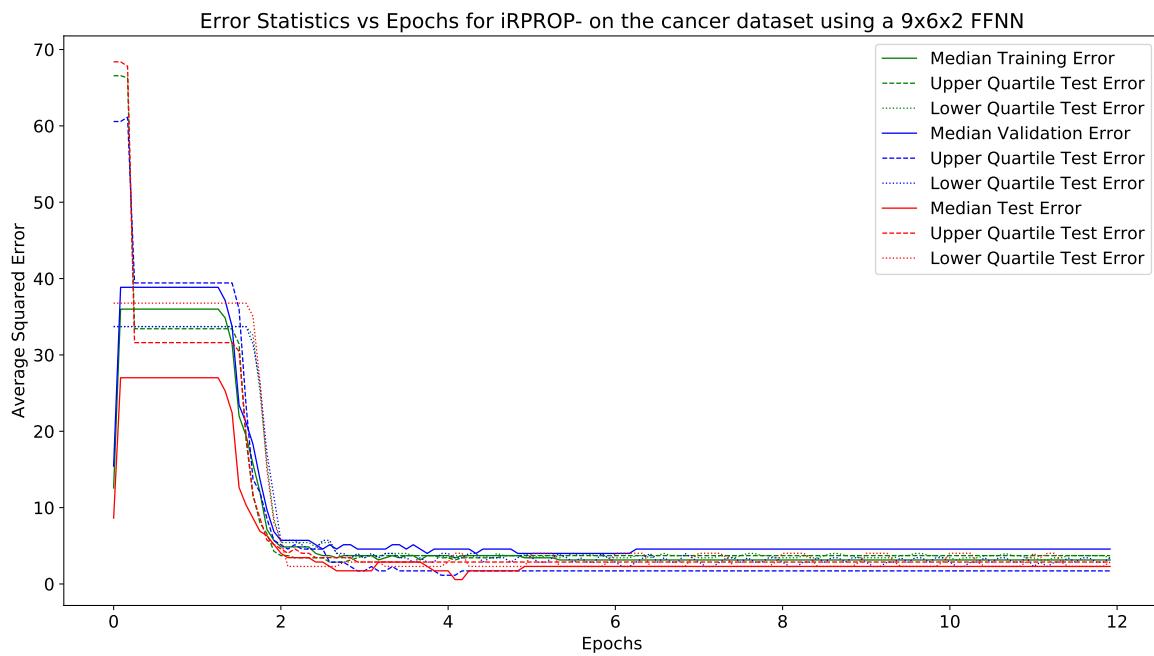
This section displays the results obtained using the cancer algorithm.

#### 9 × 6 × 2 Architecture:

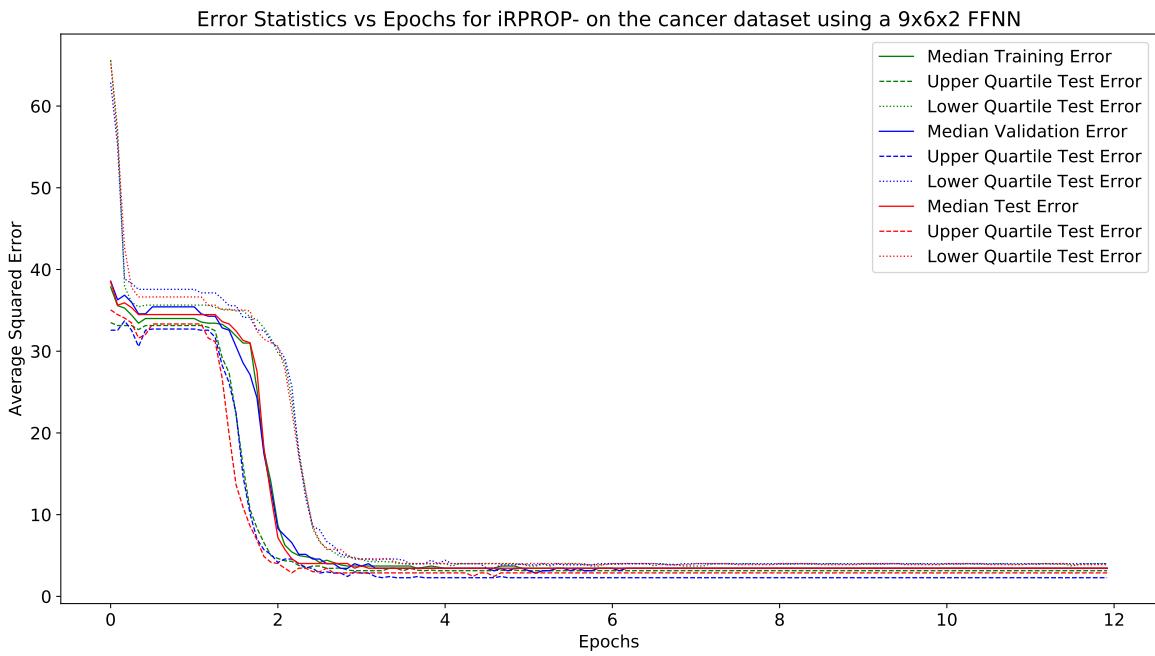
Fig. 116 shows the average and standard deviation of the test, training and validation errors. Fig. 117 shows the median and quartiles of the test, training and validation errors based on the best epoch. Finally, Fig. 118 shows the median and quartiles of the test, training and validation errors per mini-batch.



**Figure 116.** Graph of mean and standard deviation of errors vs epochs



**Figure 117.** Graph of test error vs epochs for the gradient based algorithms



**Figure 118.** Graph of test error vs epochs for the gradient based algorithms

### 16.1.22 *iRPROP+*

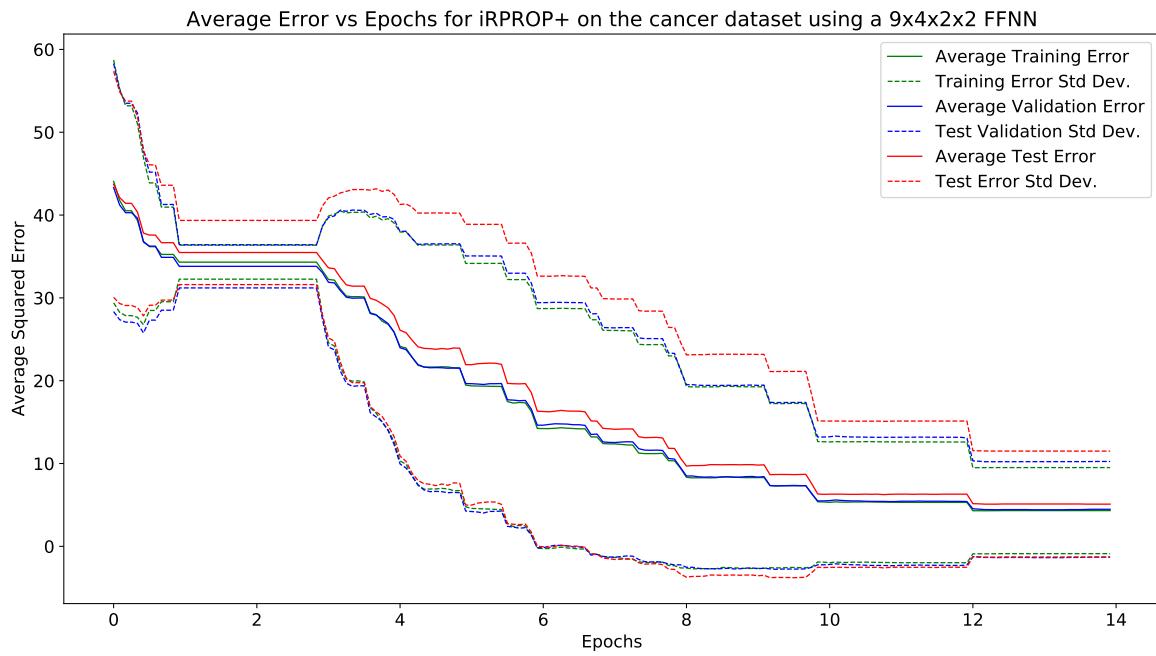
This section displays the results obtained using the iRPROP+ algorithm.

### 16.1.23 *cancer*

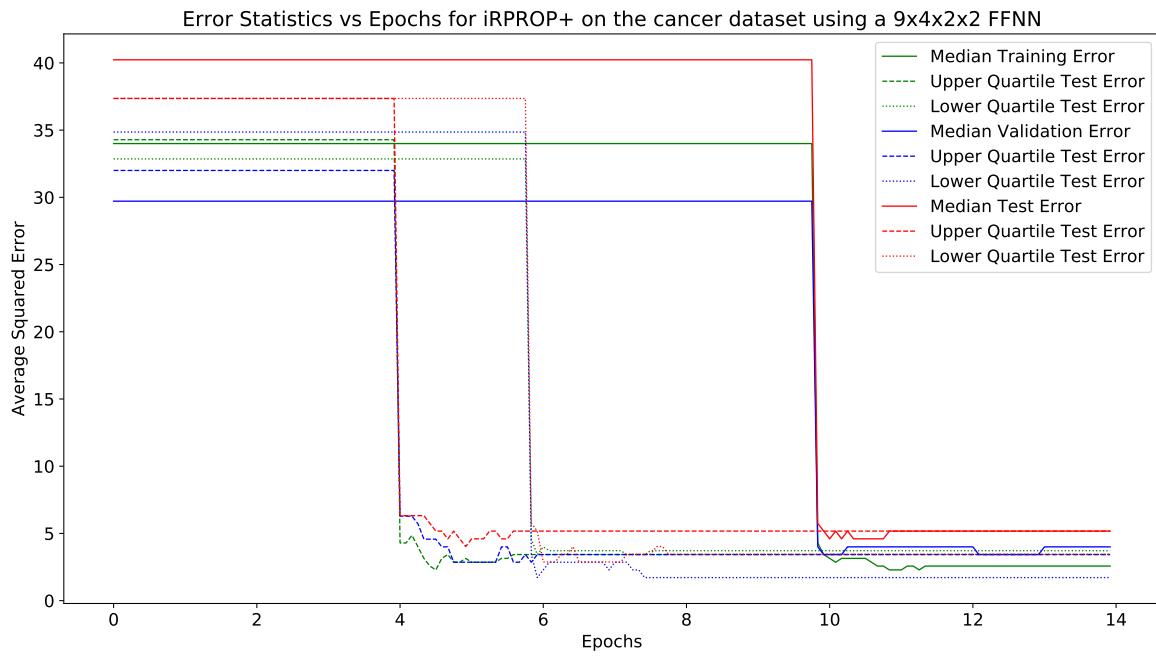
This section displays the results obtained using the cancer algorithm.

#### **9 × 4 × 2 × 2 Architecture:**

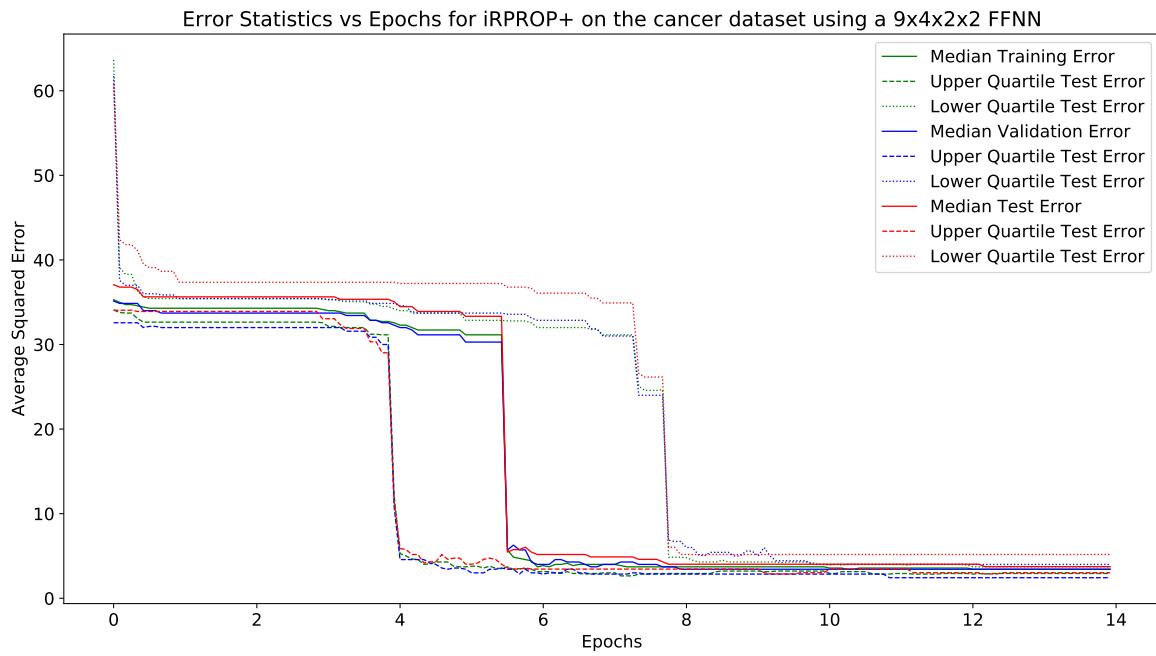
Fig. 119 shows the average and standard deviation of the test, training and validation errors. Fig. 120 shows the median and quartiles of the test, training and validation errors based on the best epoch. Finally, Fig. 121 shows the median and quartiles of the test, training and validation errors per mini-batch.



**Figure 119.** Graph of mean and standard deviation of errors vs epochs



**Figure 120.** Graph of test error vs epochs for the gradient based algorithms



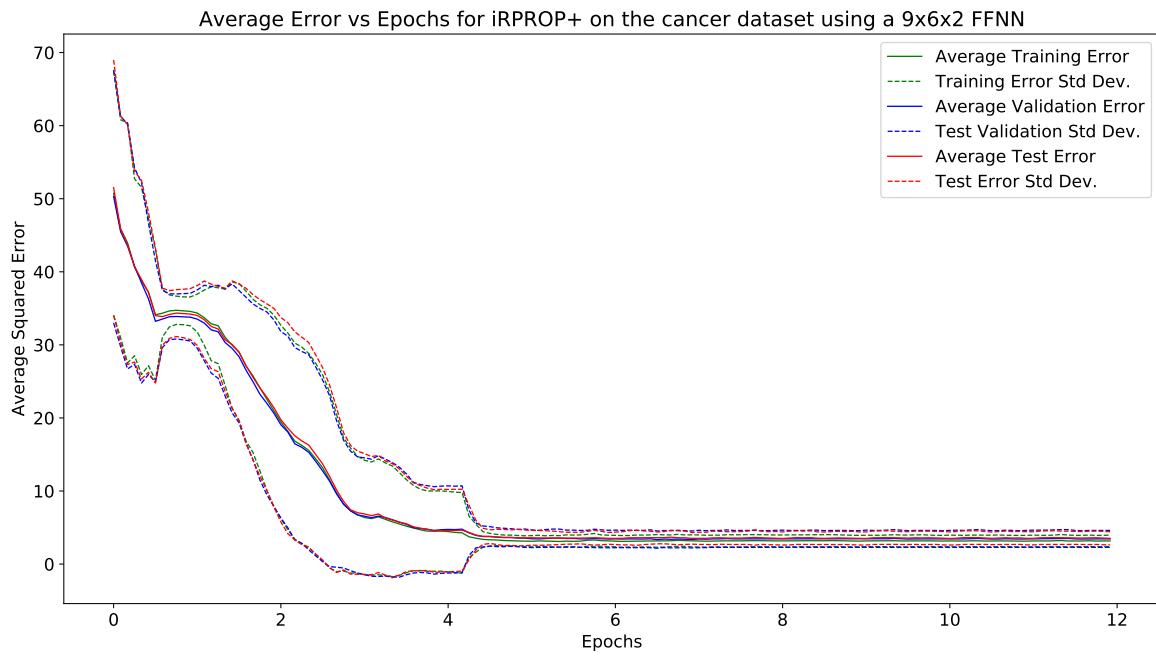
**Figure 121.** Graph of test error vs epochs for the gradient based algorithms

### 16.1.24 cancer

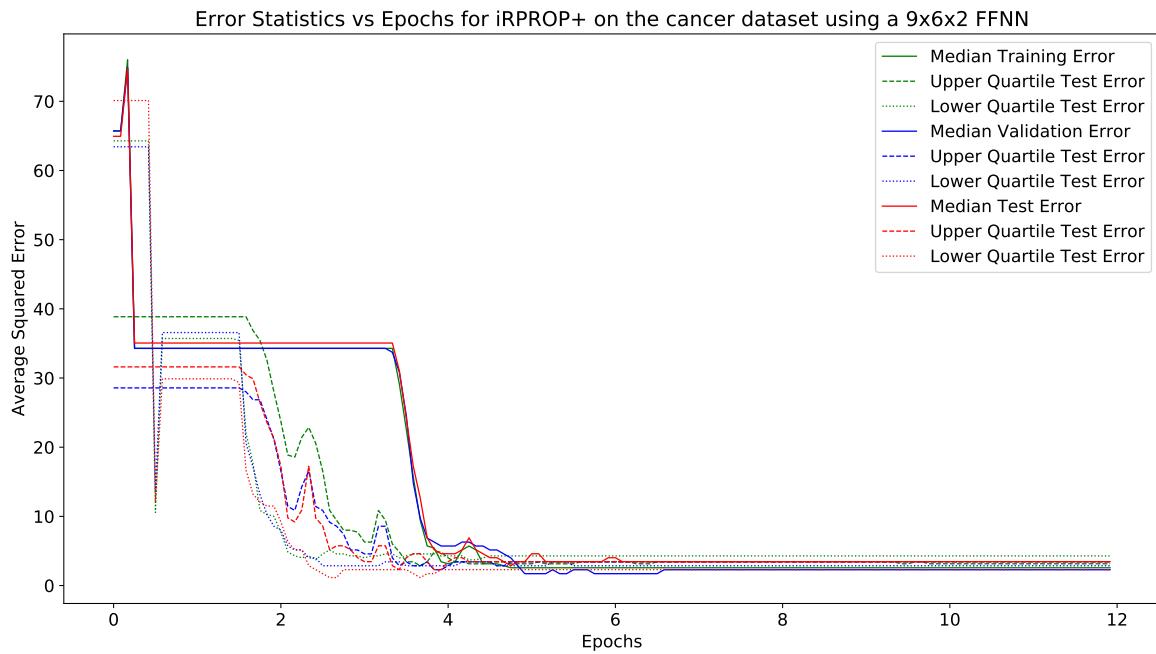
This section displays the results obtained using the cancer algorithm.

#### 9 × 6 × 2 Architecture:

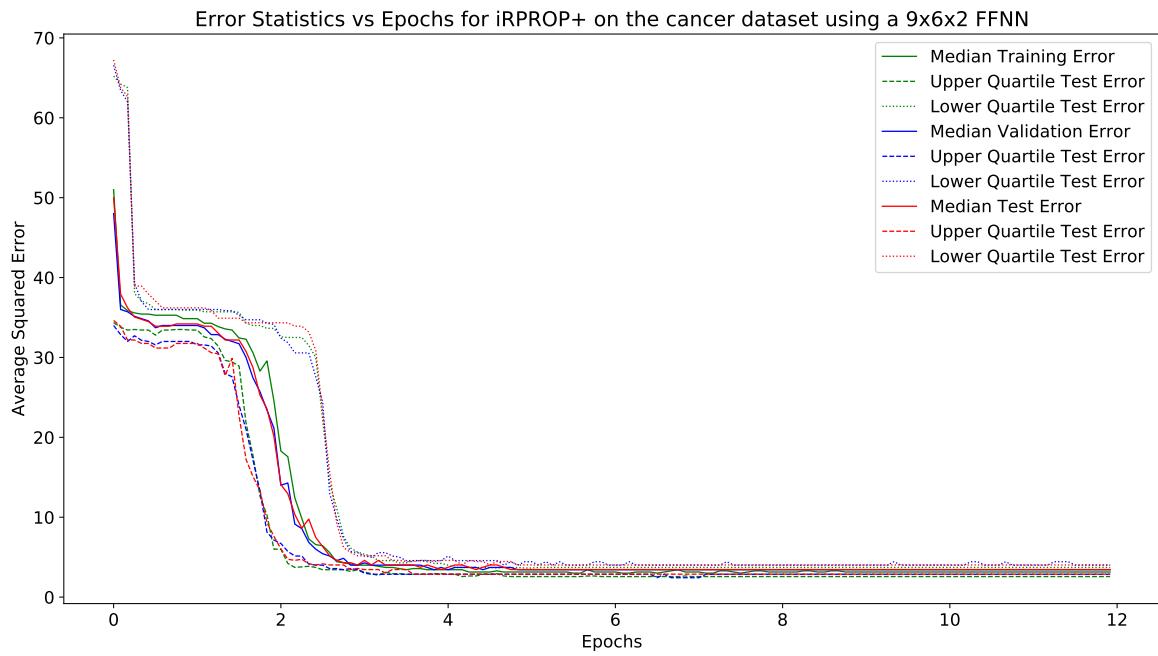
Fig. 122 shows the average and standard deviation of the test, training and validation errors. Fig. 123 shows the median and quartiles of the test, training and validation errors based on the best epoch. Finally, Fig. 124 shows the median and quartiles of the test, training and validation errors per mini-batch.



**Figure 122.** Graph of mean and standard deviation of errors vs epochs



**Figure 123.** Graph of test error vs epochs for the gradient based algorithms



**Figure 124.** Graph of test error vs epochs for the gradient based algorithms

### 16.1.25 PSO

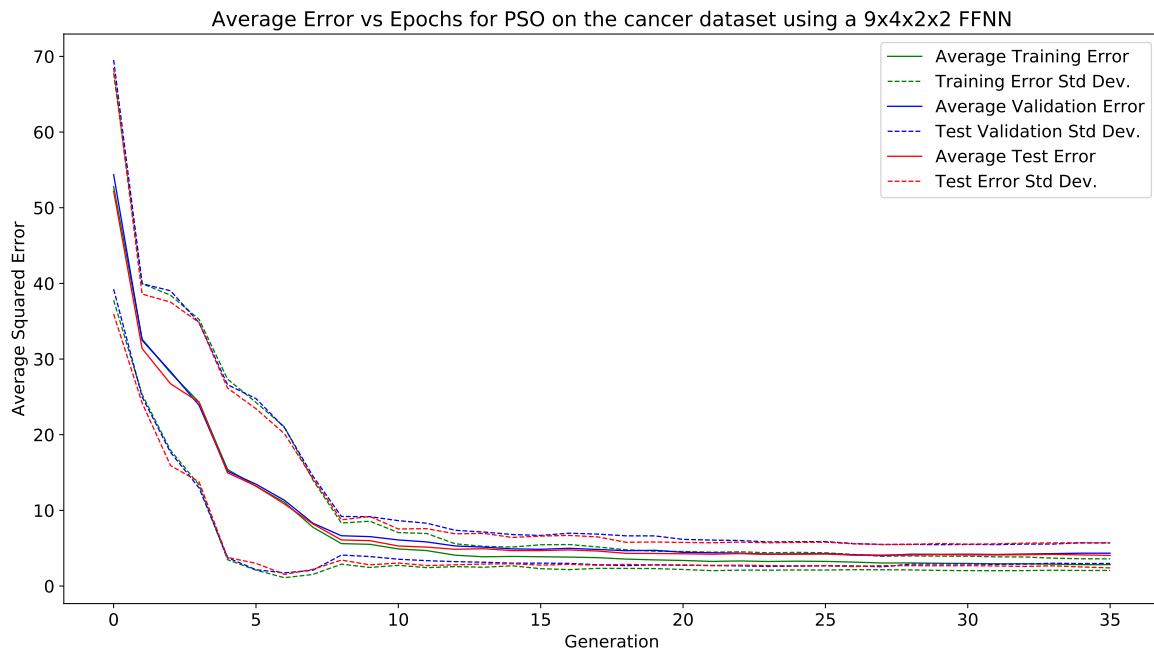
This section displays the results obtained using the PSO algorithm.

### 16.1.26 cancer

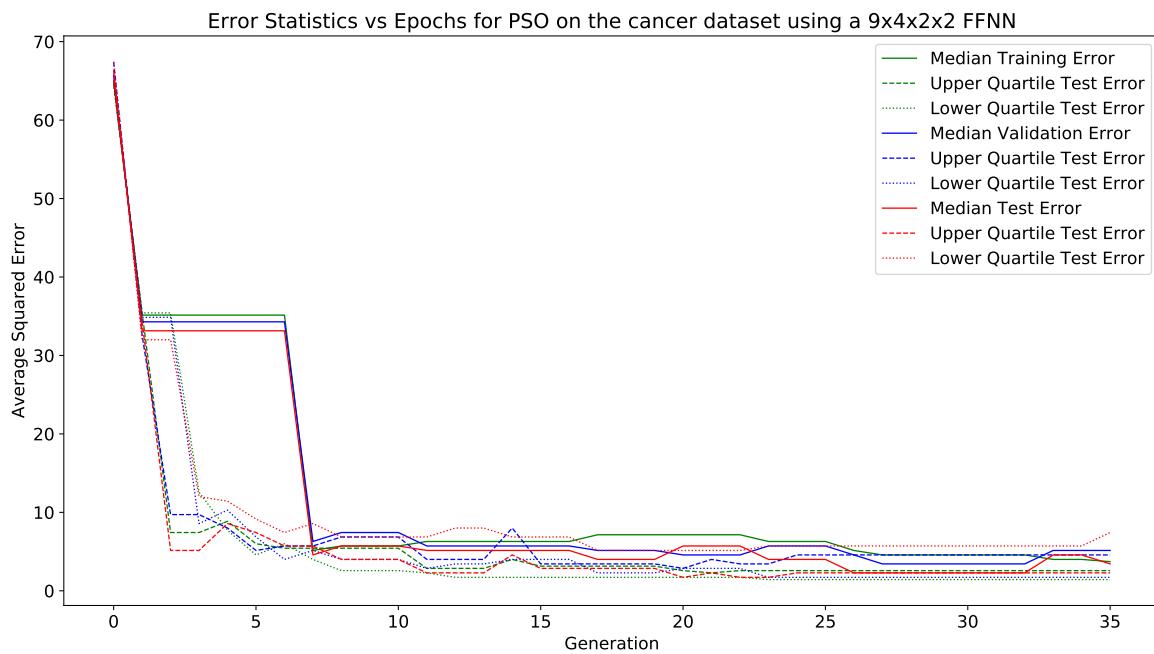
This section displays the results obtained using the cancer algorithm.

#### 9 × 4 × 2 × 2 Architecture:

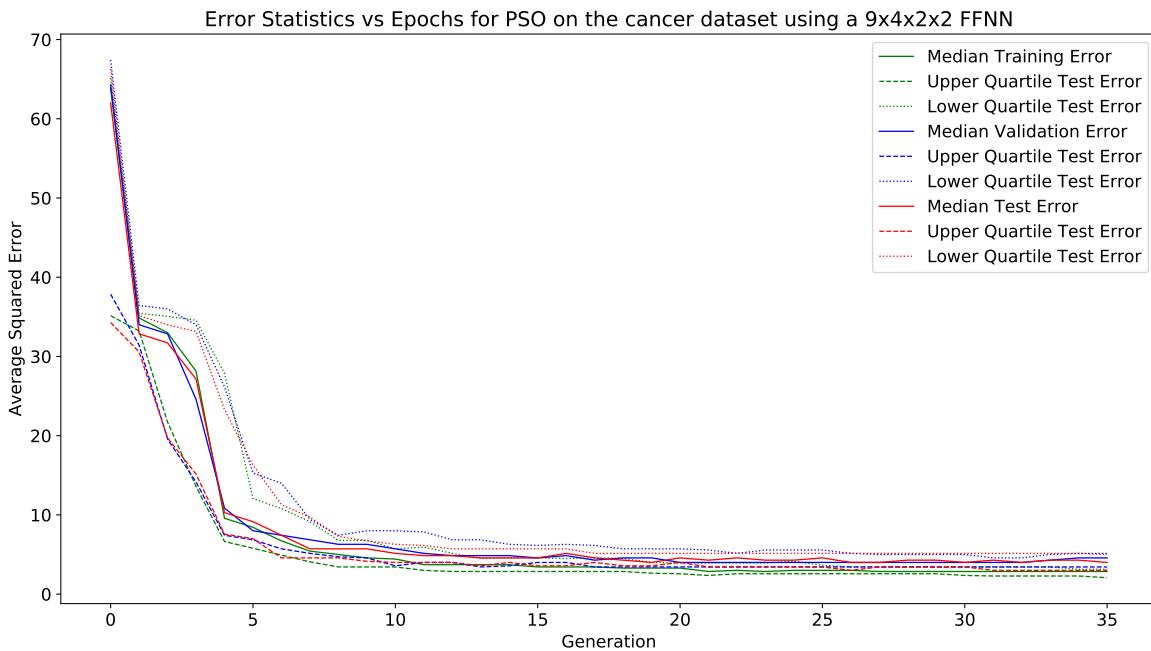
Fig. 125 shows the average and standard deviation of the test, training and validation errors. Fig. 126 shows the median and quartiles of the test, training and validation errors based on the best epoch. Finally, Fig. 127 shows the median and quartiles of the test, training and validation errors per mini-batch.



**Figure 125.** Graph of mean and standard deviation of errors vs epochs



**Figure 126.** Graph of test error vs epochs for the gradient based algorithms



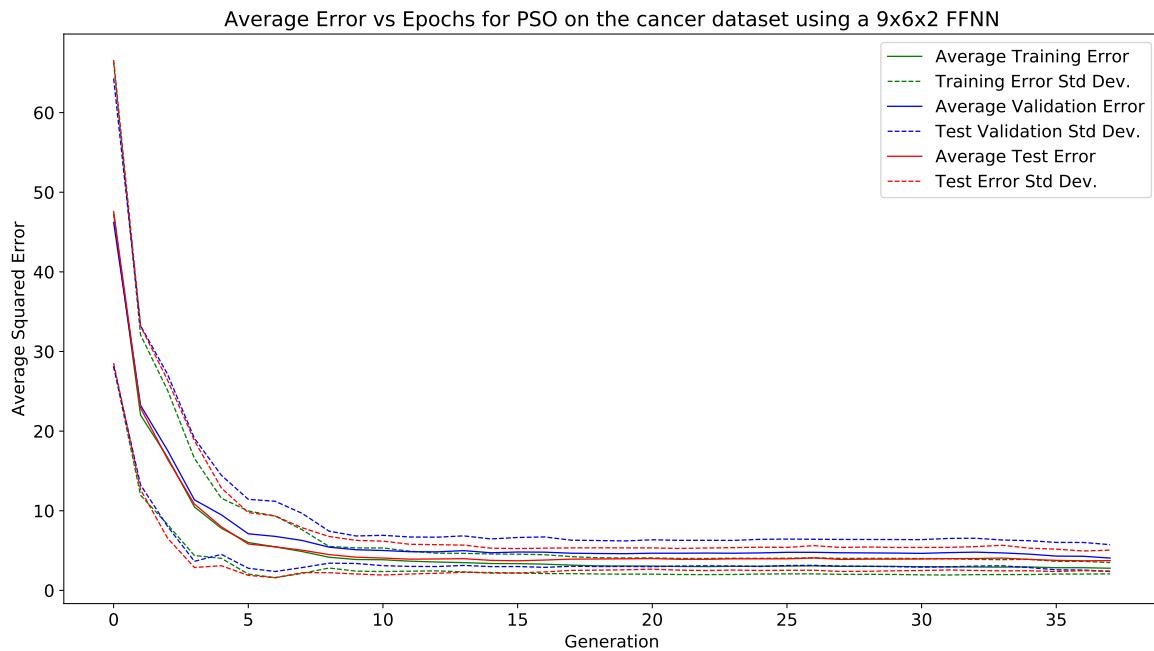
**Figure 127.** Graph of test error vs epochs for the gradient based algorithms

### 16.1.27 cancer

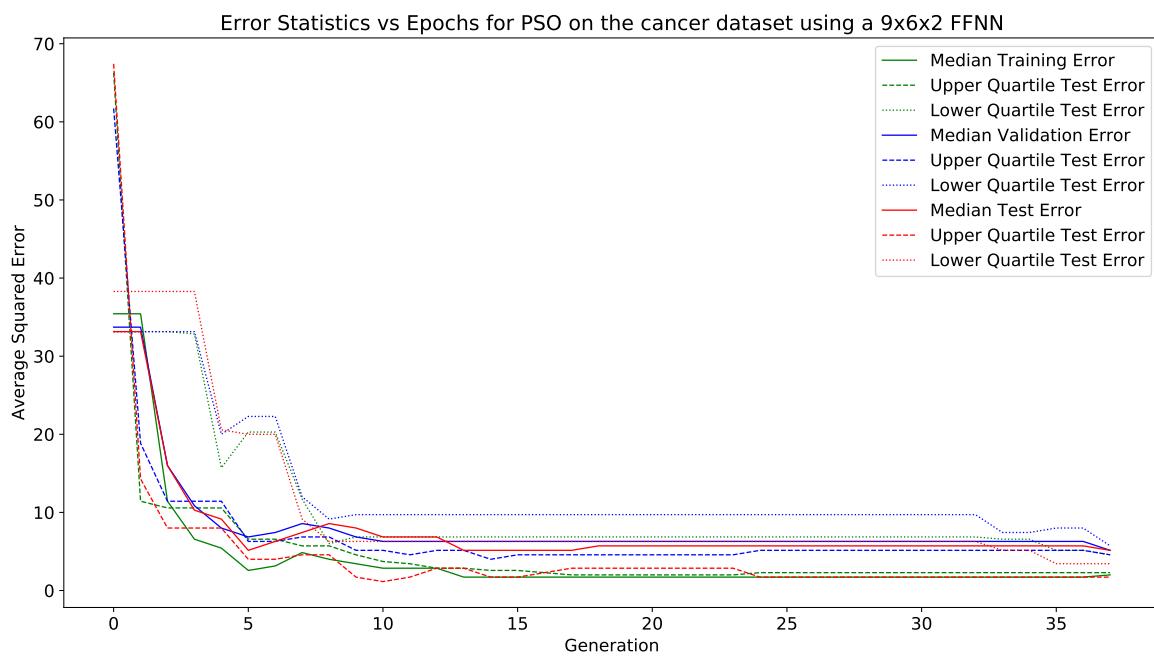
This section displays the results obtained using the cancer algorithm.

#### 9 × 6 × 2 Architecture:

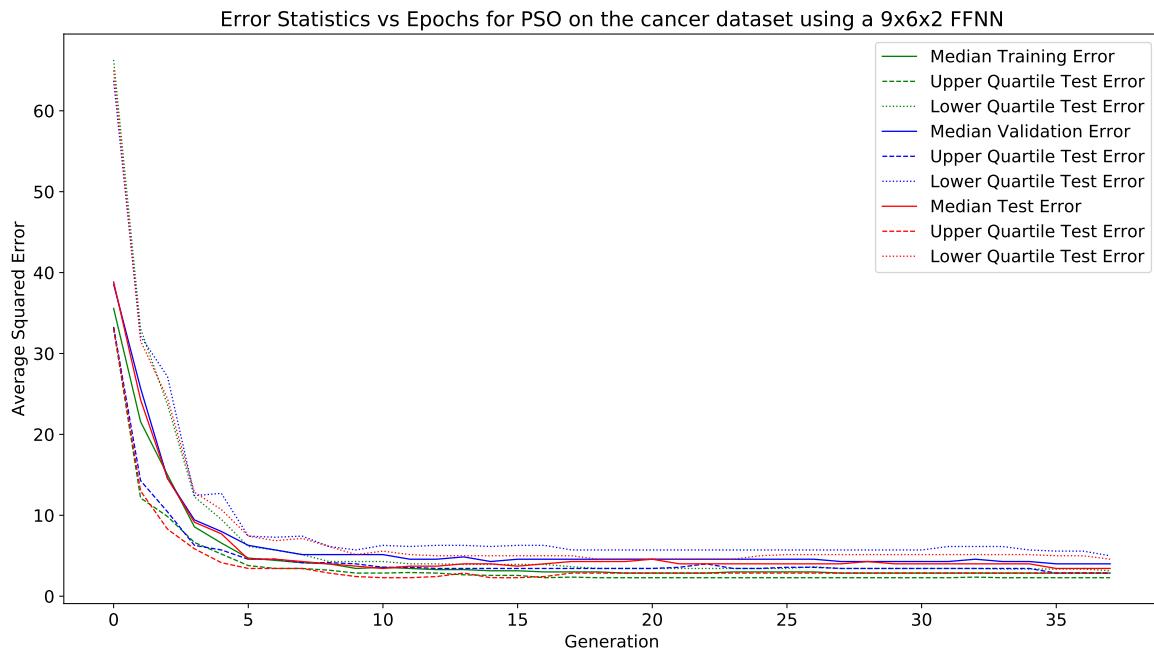
Fig. 128 shows the average and standard deviation of the test, training and validation errors. Fig. 129 shows the median and quartiles of the test, training and validation errors based on the best epoch. Finally, Fig. 130 shows the median and quartiles of the test, training and validation errors per mini-batch.



**Figure 128.** Graph of mean and standard deviation of errors vs epochs



**Figure 129.** Graph of test error vs epochs for the gradient based algorithms



**Figure 130.** Graph of test error vs epochs for the gradient based algorithms

### 16.1.28 QuickProp

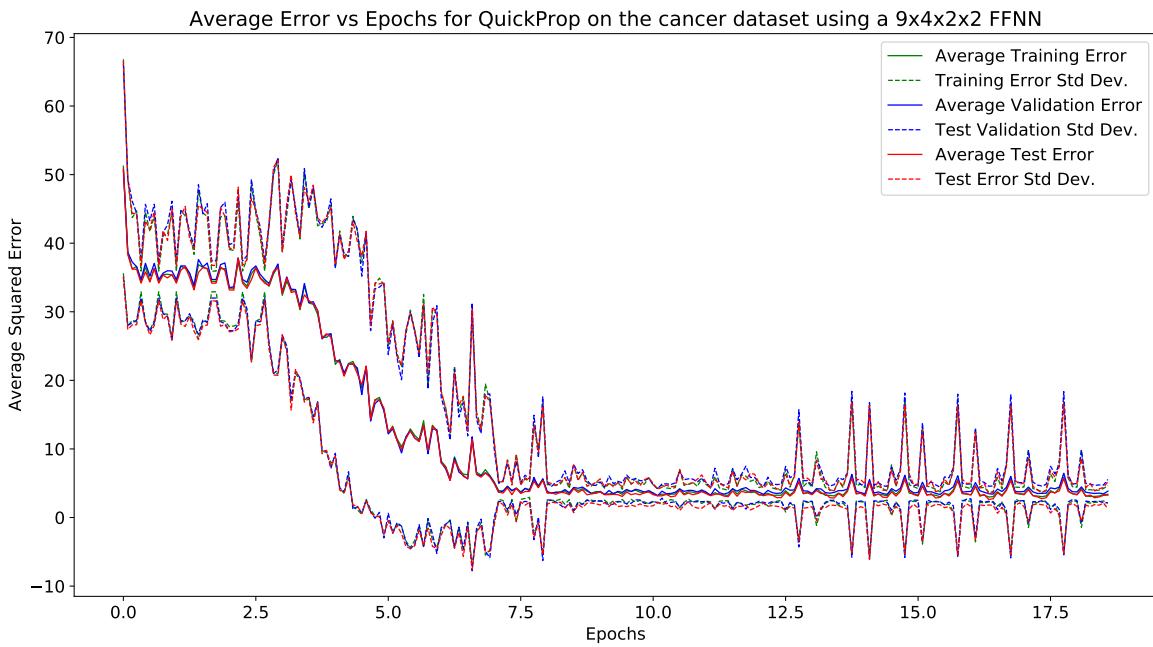
This section displays the results obtained using the QuickProp algorithm.

### 16.1.29 cancer

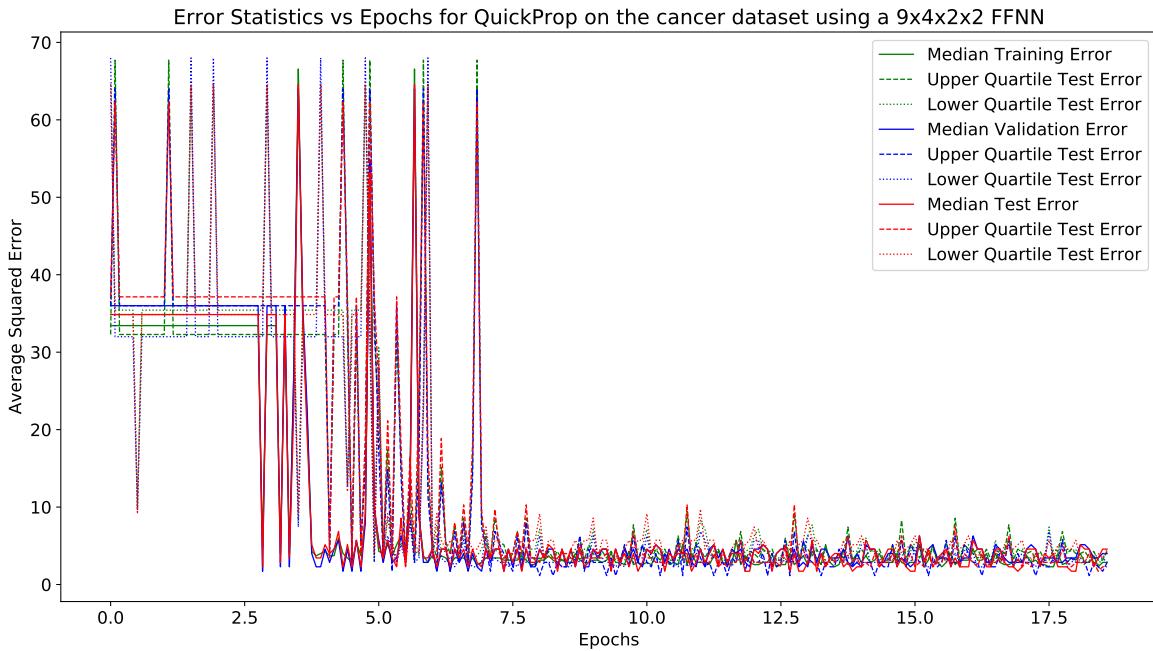
This section displays the results obtained using the cancer algorithm.

#### 9 × 4 × 2 × 2 Architecture:

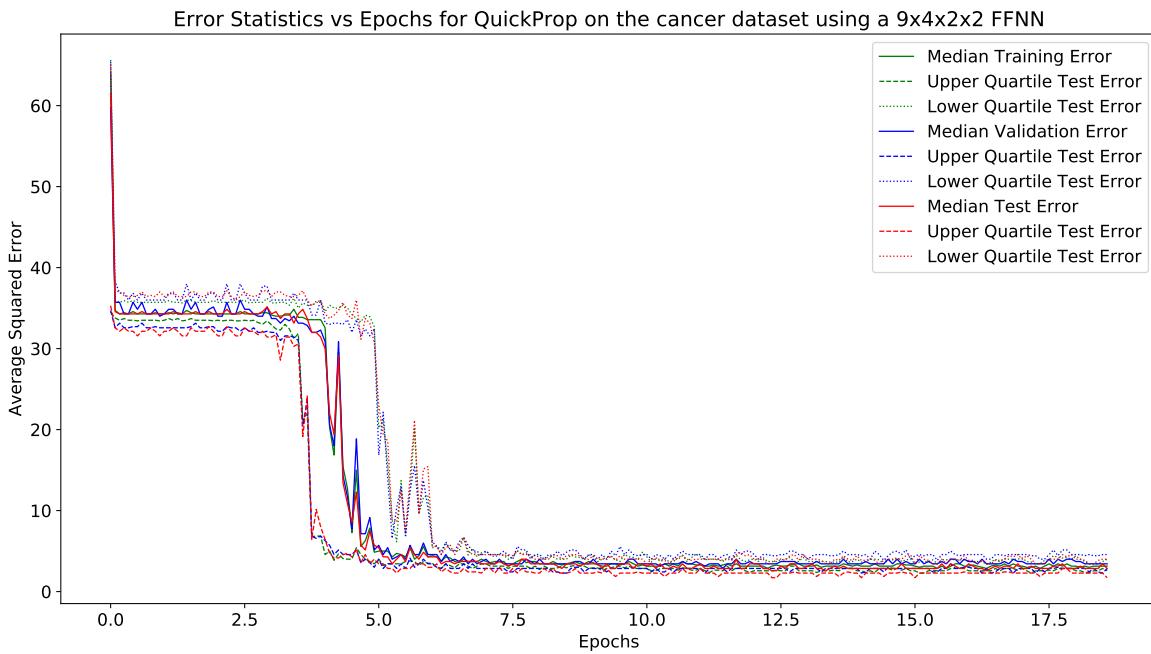
Fig. 131 shows the average and standard deviation of the test, training and validation errors. Fig. 132 shows the median and quartiles of the test, training and validation errors based on the best epoch. Finally, Fig. 133 shows the median and quartiles of the test, training and validation errors per mini-batch.



**Figure 131.** Graph of mean and standard deviation of errors vs epochs



**Figure 132.** Graph of test error vs epochs for the gradient based algorithms



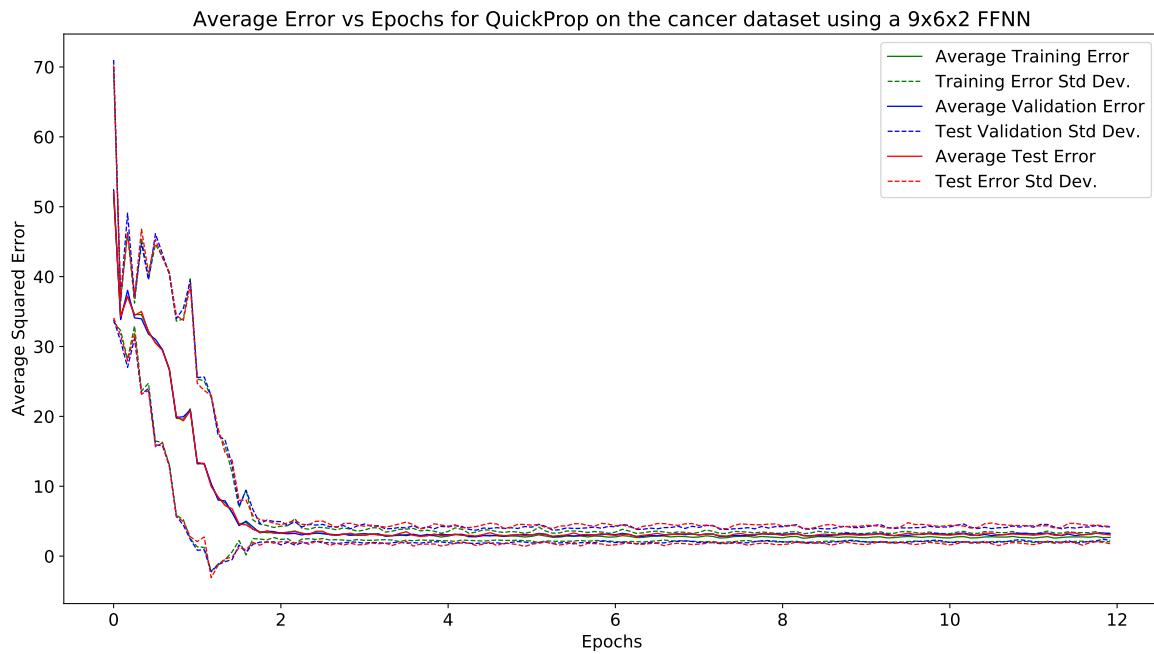
**Figure 133.** Graph of test error vs epochs for the gradient based algorithms

### 16.1.30 cancer

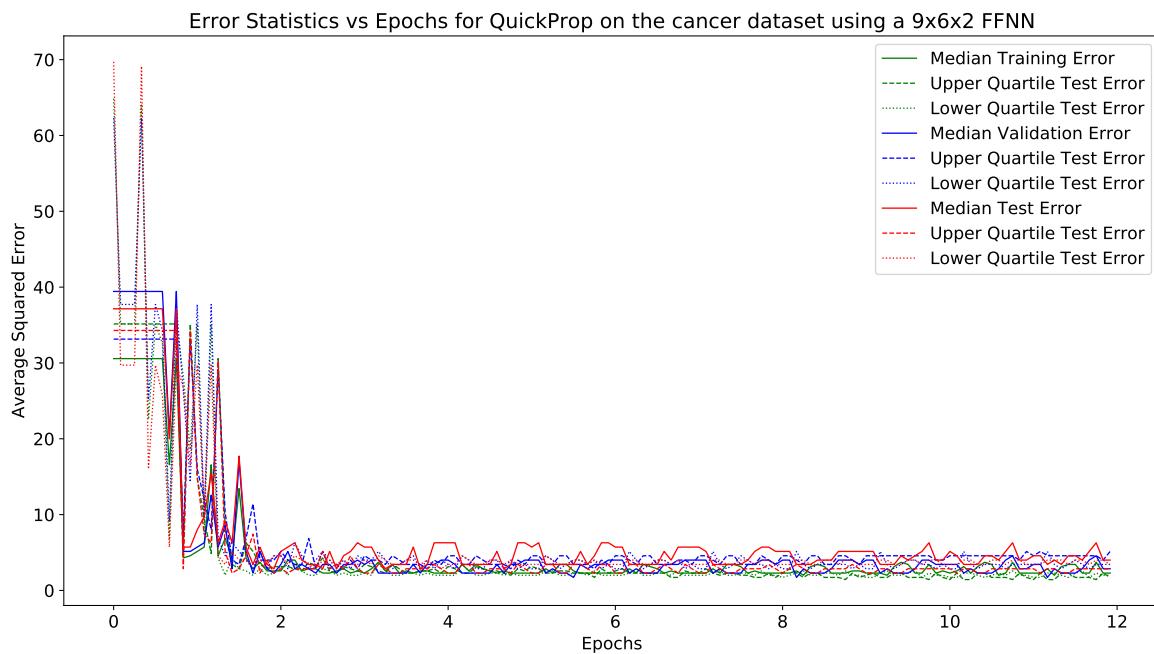
This section displays the results obtained using the cancer algorithm.

#### 9 × 6 × 2 Architecture:

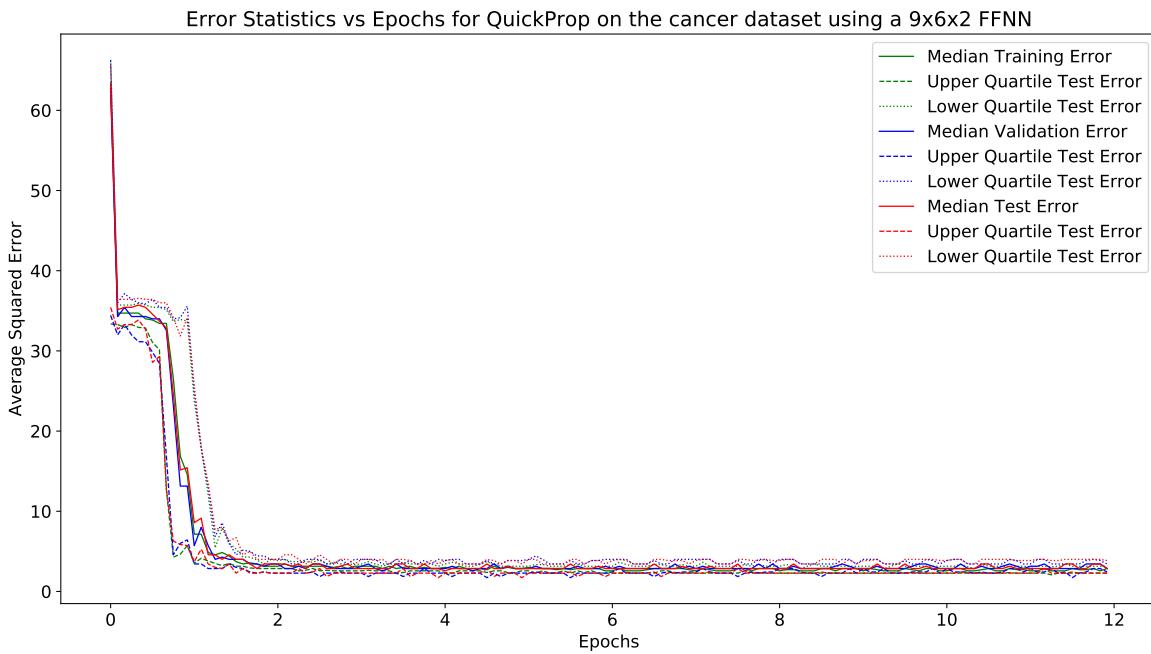
Fig. 134 shows the average and standard deviation of the test, training and validation errors. Fig. 135 shows the median and quartiles of the test, training and validation errors based on the best epoch. Finally, Fig. 136 shows the median and quartiles of the test, training and validation errors per mini-batch.



**Figure 134.** Graph of mean and standard deviation of errors vs epochs



**Figure 135.** Graph of test error vs epochs for the gradient based algorithms



**Figure 136.** Graph of test error vs epochs for the gradient based algorithms

### 16.1.31 RPROP-

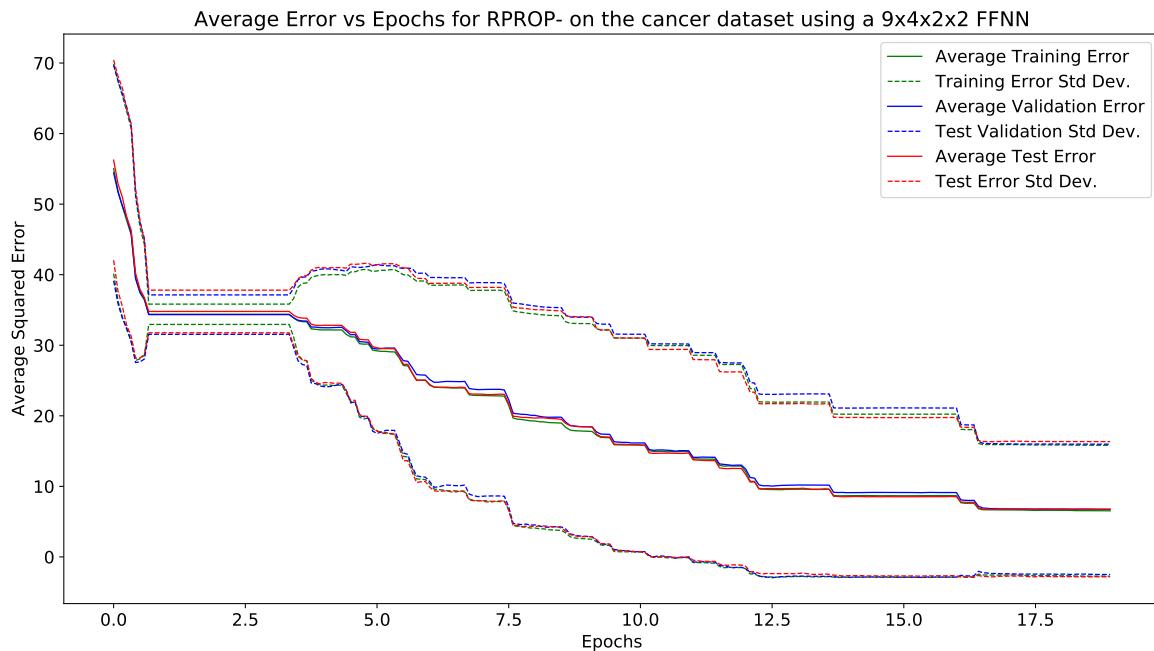
This section displays the results obtained using the RPROP- algorithm.

### 16.1.32 cancer

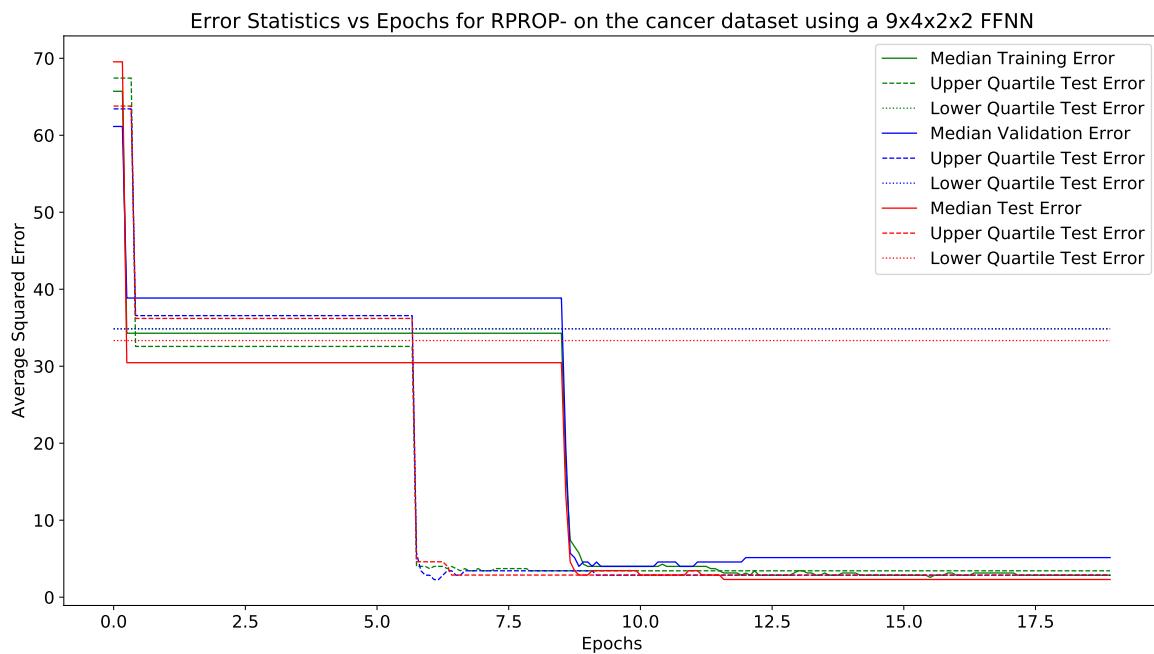
This section displays the results obtained using the cancer algorithm.

#### 9 × 4 × 2 × 2 Architecture:

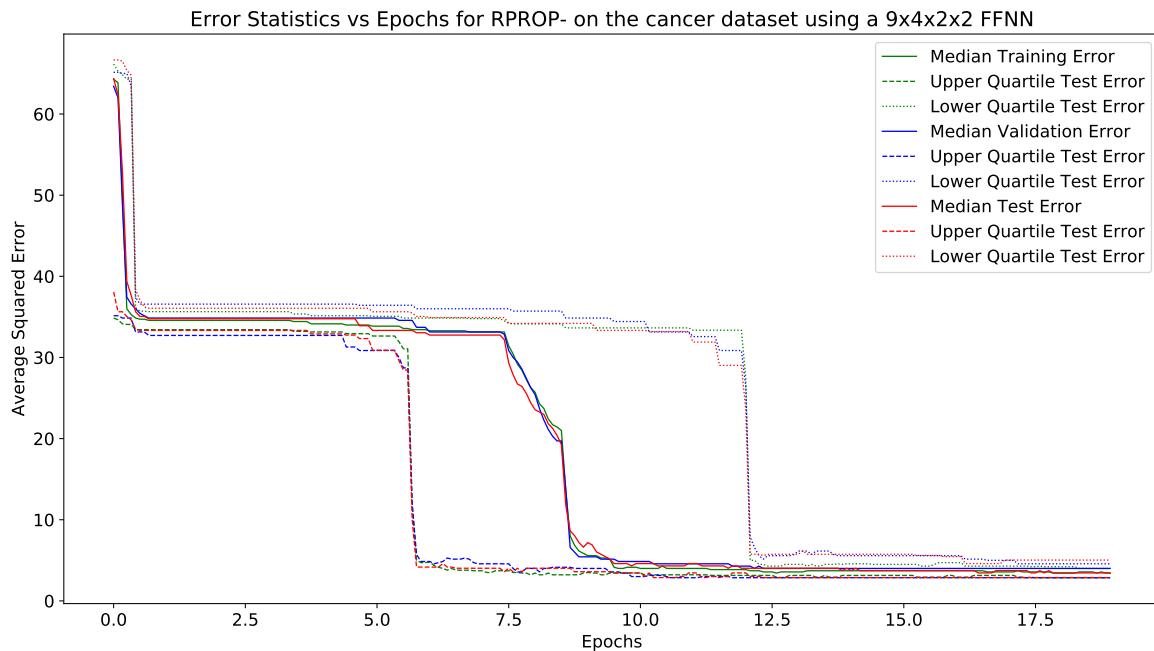
Fig. 137 shows the average and standard deviation of the test, training and validation errors. Fig. 138 shows the median and quartiles of the test, training and validation errors based on the best epoch. Finally, Fig. 139 shows the median and quartiles of the test, training and validation errors per mini-batch.



**Figure 137.** Graph of mean and standard deviation of errors vs epochs



**Figure 138.** Graph of test error vs epochs for the gradient based algorithms



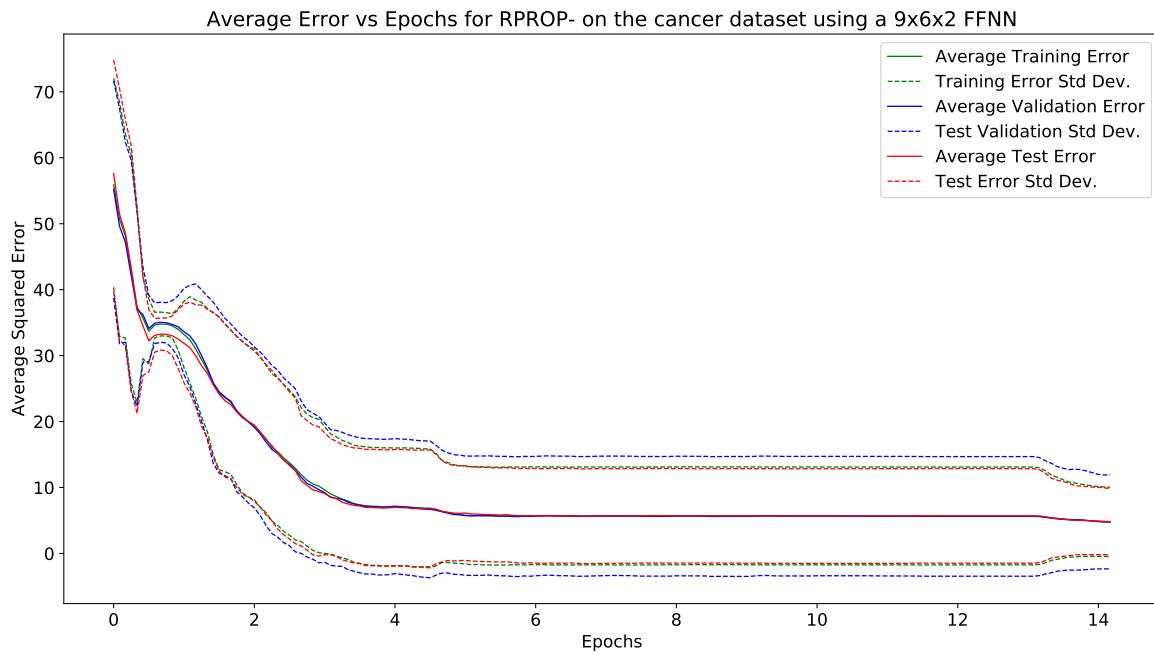
**Figure 139.** Graph of test error vs epochs for the gradient based algorithms

### 16.1.33 cancer

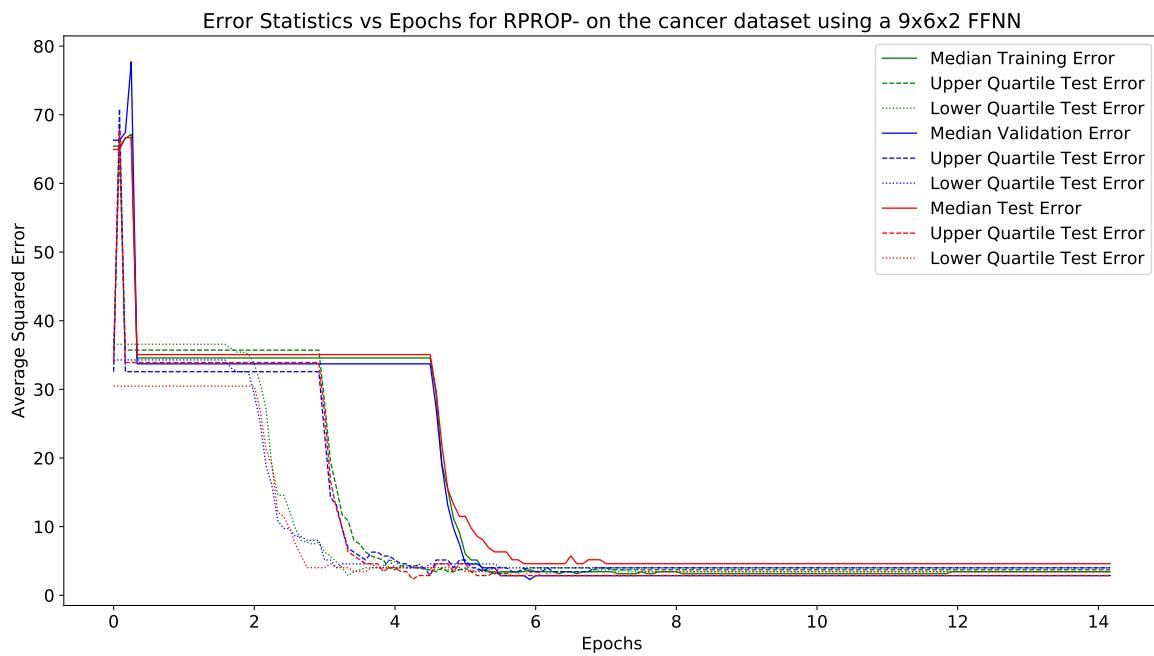
This section displays the results obtained using the cancer algorithm.

#### 9 × 6 × 2 Architecture:

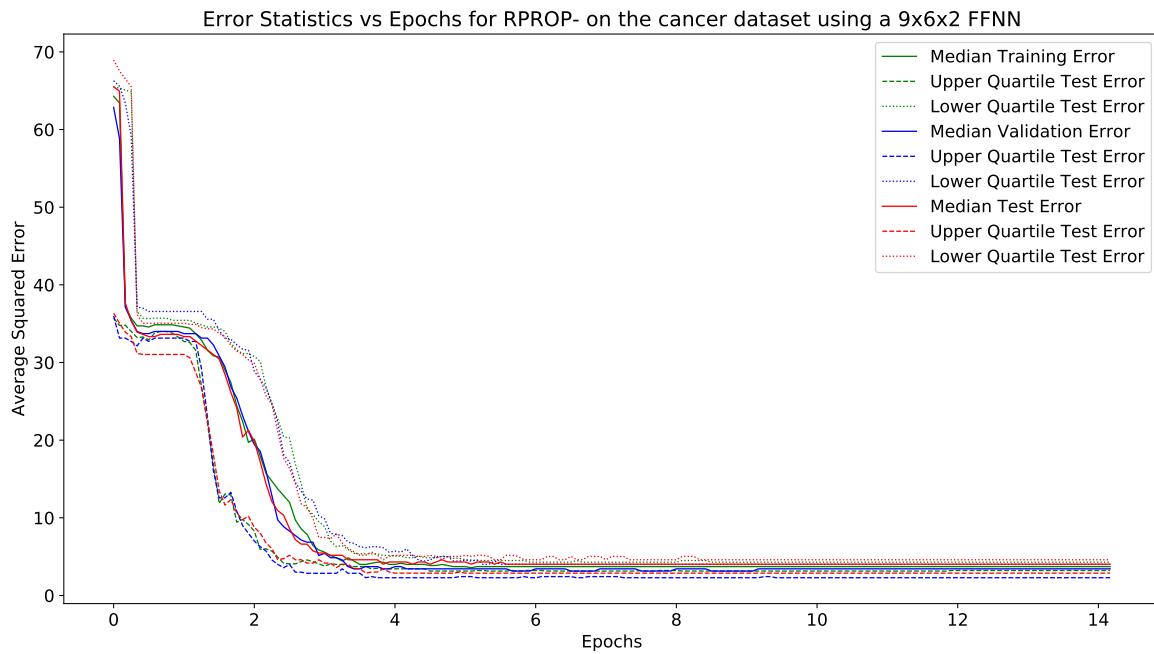
Fig. 140 shows the average and standard deviation of the test, training and validation errors. Fig. 141 shows the median and quartiles of the test, training and validation errors based on the best epoch. Finally, Fig. 142 shows the median and quartiles of the test, training and validation errors per mini-batch.



**Figure 140.** Graph of mean and standard deviation of errors vs epochs



**Figure 141.** Graph of test error vs epochs for the gradient based algorithms



**Figure 142.** Graph of test error vs epochs for the gradient based algorithms

### 16.1.34 RPROP+

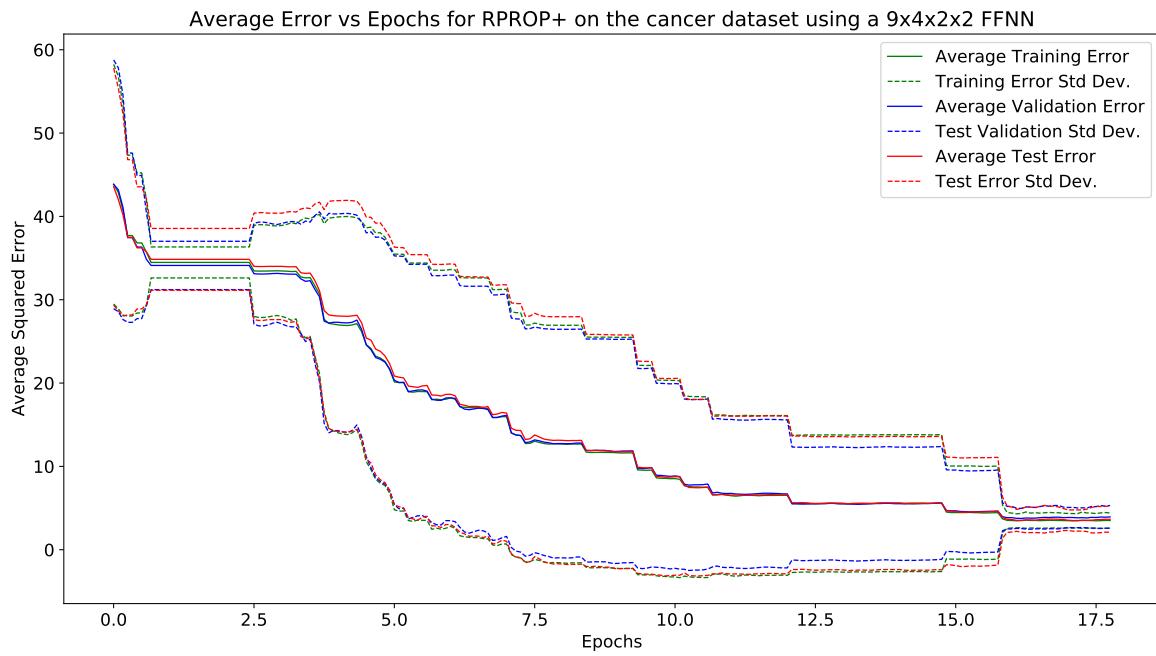
This section displays the results obtained using the RPROP+ algorithm.

### 16.1.35 cancer

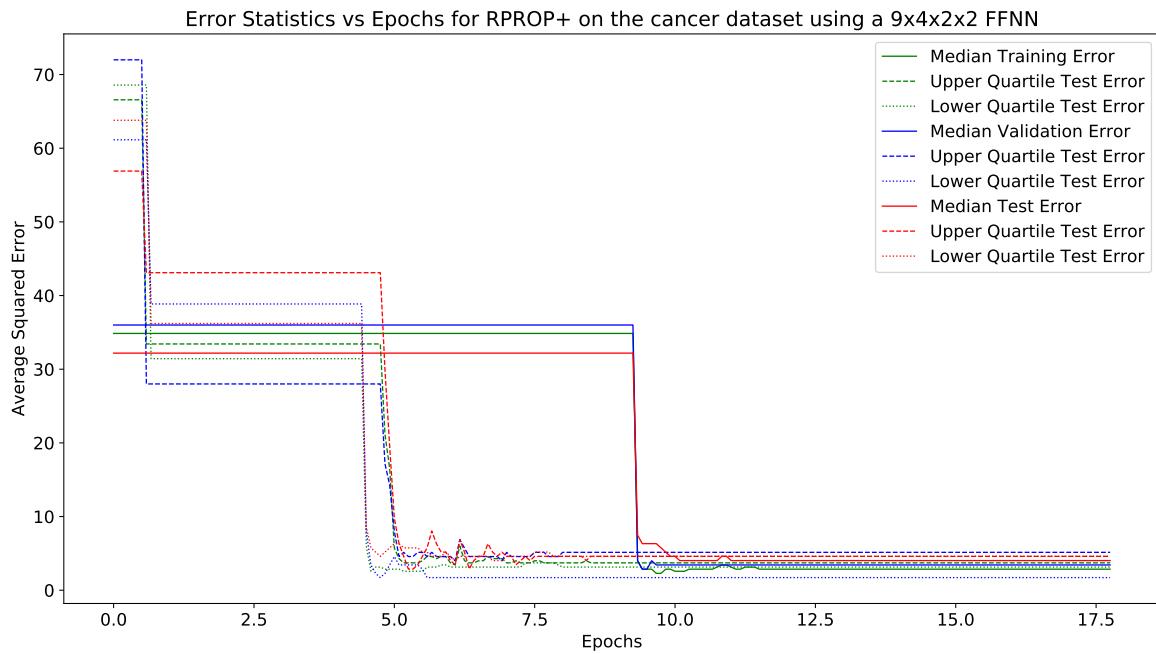
This section displays the results obtained using the cancer algorithm.

#### 9 × 4 × 2 × 2 Architecture:

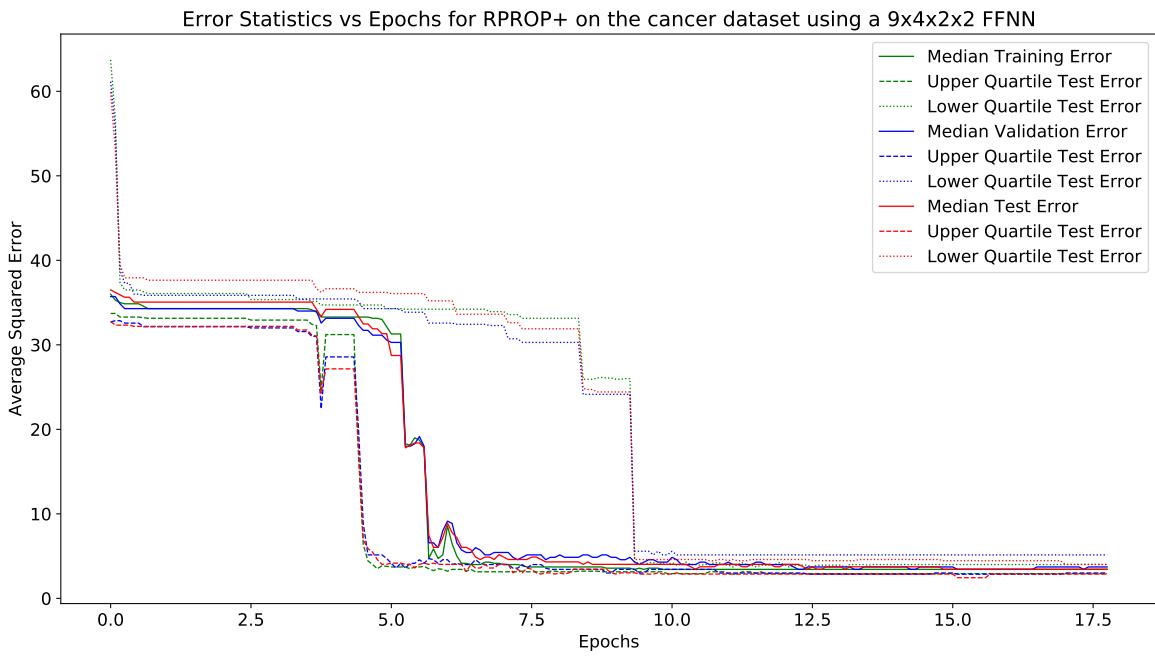
Fig. 143 shows the average and standard deviation of the test, training and validation errors. Fig. 144 shows the median and quartiles of the test, training and validation errors based on the best epoch. Finally, Fig. 145 shows the median and quartiles of the test, training and validation errors per mini-batch.



**Figure 143.** Graph of mean and standard deviation of errors vs epochs



**Figure 144.** Graph of test error vs epochs for the gradient based algorithms



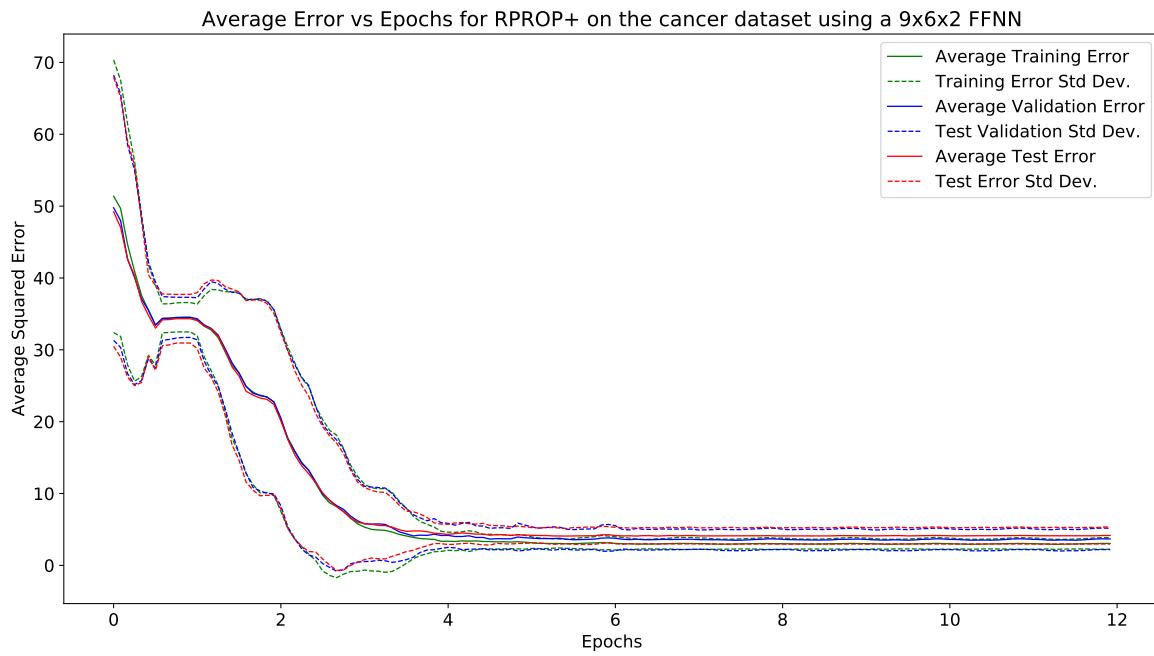
**Figure 145.** Graph of test error vs epochs for the gradient based algorithms

### 16.1.36 cancer

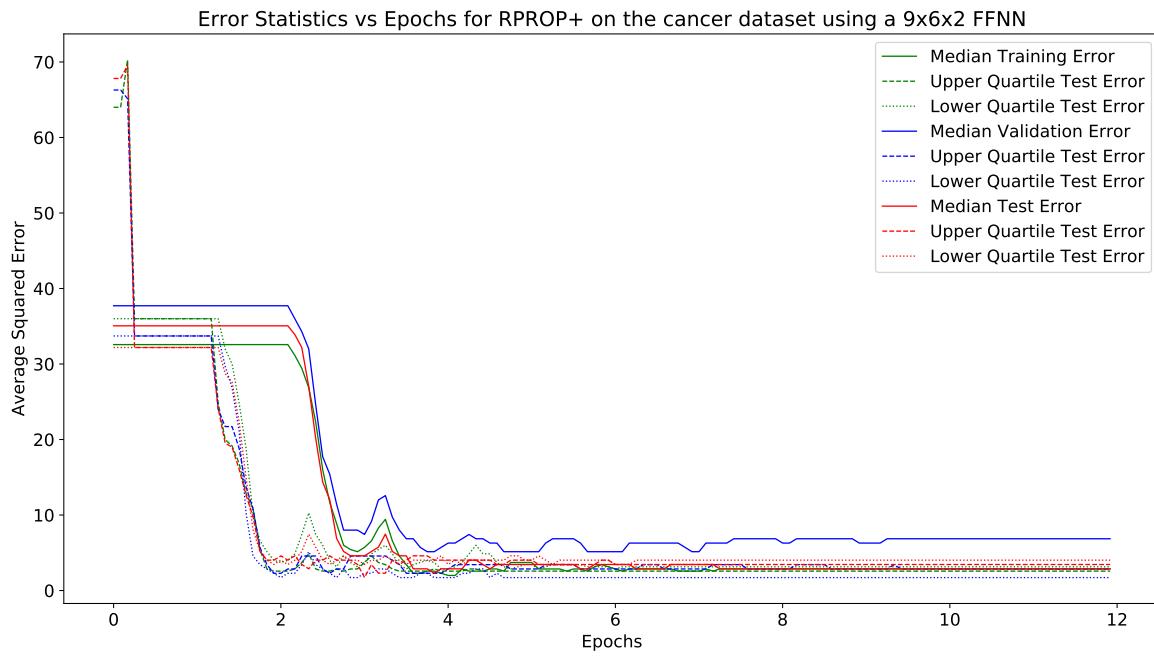
This section displays the results obtained using the cancer algorithm.

#### 9 × 6 × 2 Architecture:

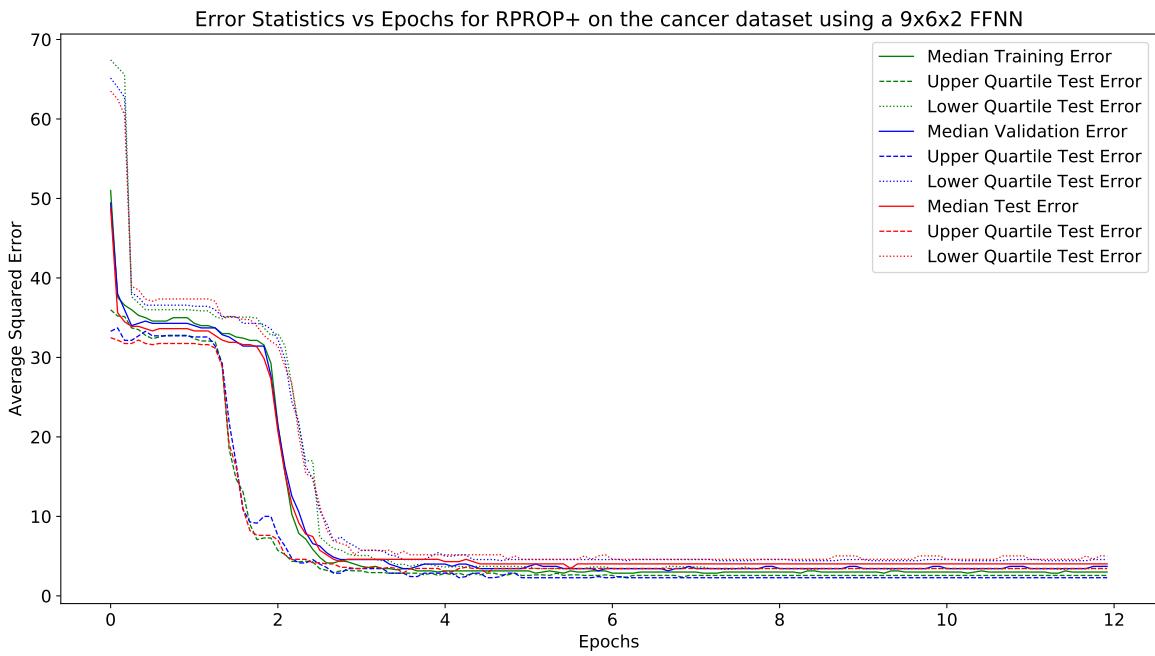
Fig. 146 shows the average and standard deviation of the test, training and validation errors. Fig. 147 shows the median and quartiles of the test, training and validation errors based on the best epoch. Finally, Fig. 148 shows the median and quartiles of the test, training and validation errors per mini-batch.



**Figure 146.** Graph of mean and standard deviation of errors vs epochs



**Figure 147.** Graph of test error vs epochs for the gradient based algorithms



**Figure 148.** Graph of test error vs epochs for the gradient based algorithms

## 16.2 card

This section displays the results obtained in the card dataset.

### 16.2.1 ADAGRAD

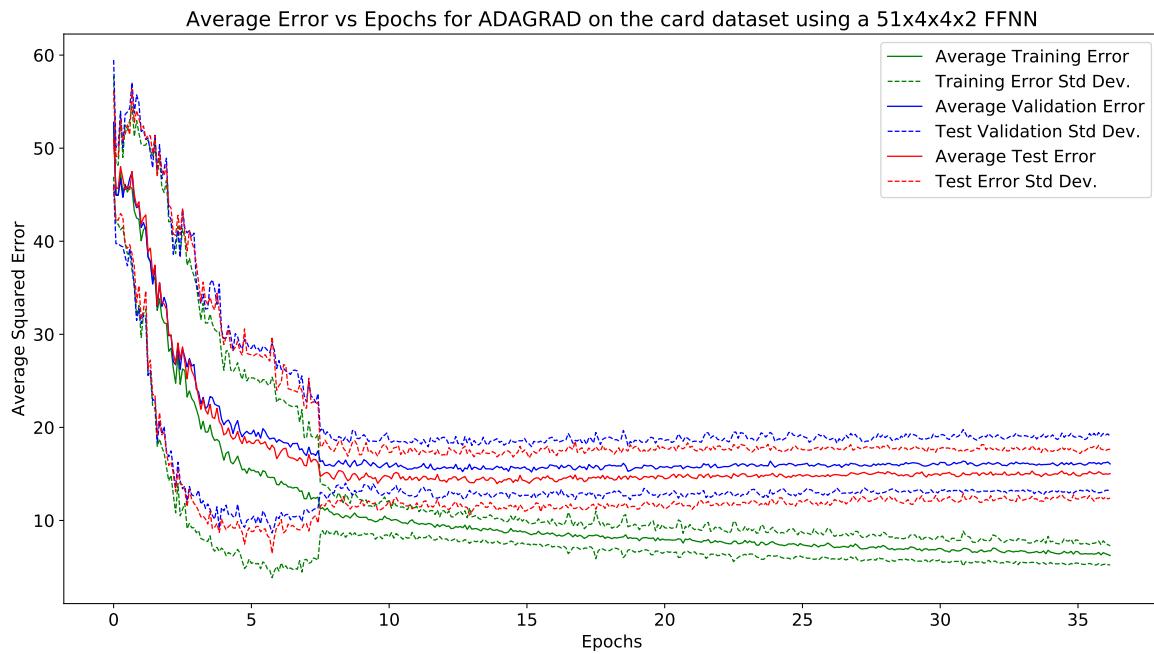
This section displays the results obtained using the ADAGRAD algorithm.

### 16.2.2 card

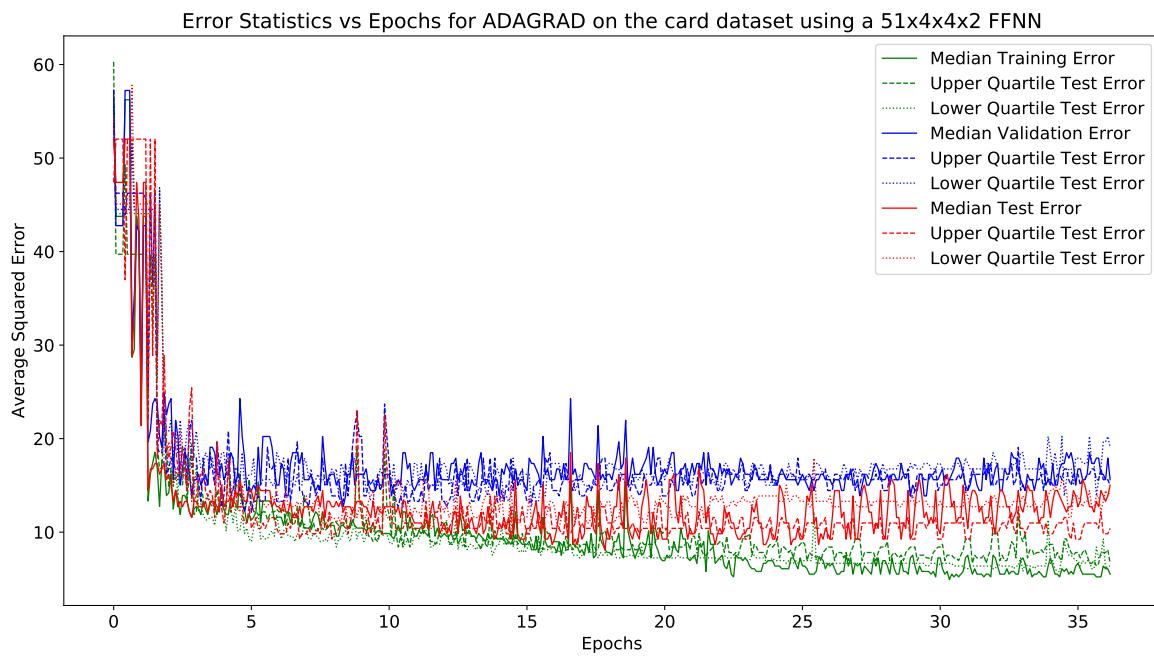
This section displays the results obtained using the card algorithm.

#### 51 × 4 × 4 × 2 Architecture:

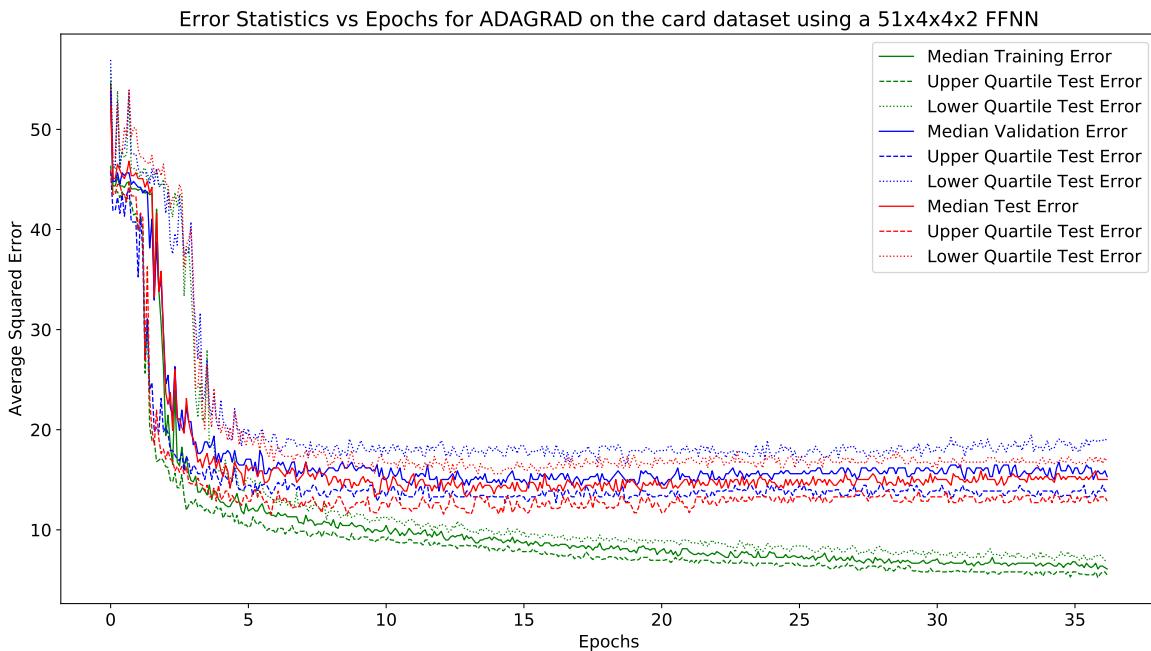
Fig. 149 shows the average and standard deviation of the test, training and validation errors. Fig. 150 shows the median and quartiles of the test, training and validation errors based on the best epoch. Finally, Fig. 151 shows the median and quartiles of the test, training and validation errors per mini-batch.



**Figure 149.** Graph of mean and standard deviation of errors vs epochs



**Figure 150.** Graph of test error vs epochs for the gradient based algorithms



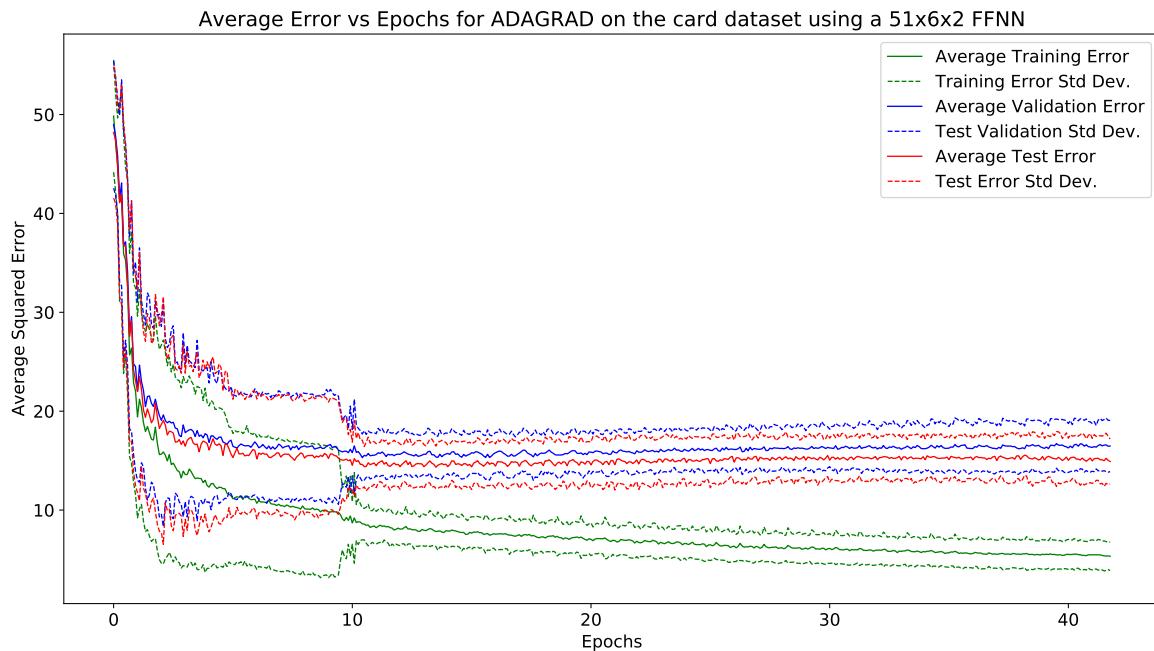
**Figure 151.** Graph of test error vs epochs for the gradient based algorithms

### 16.2.3 card

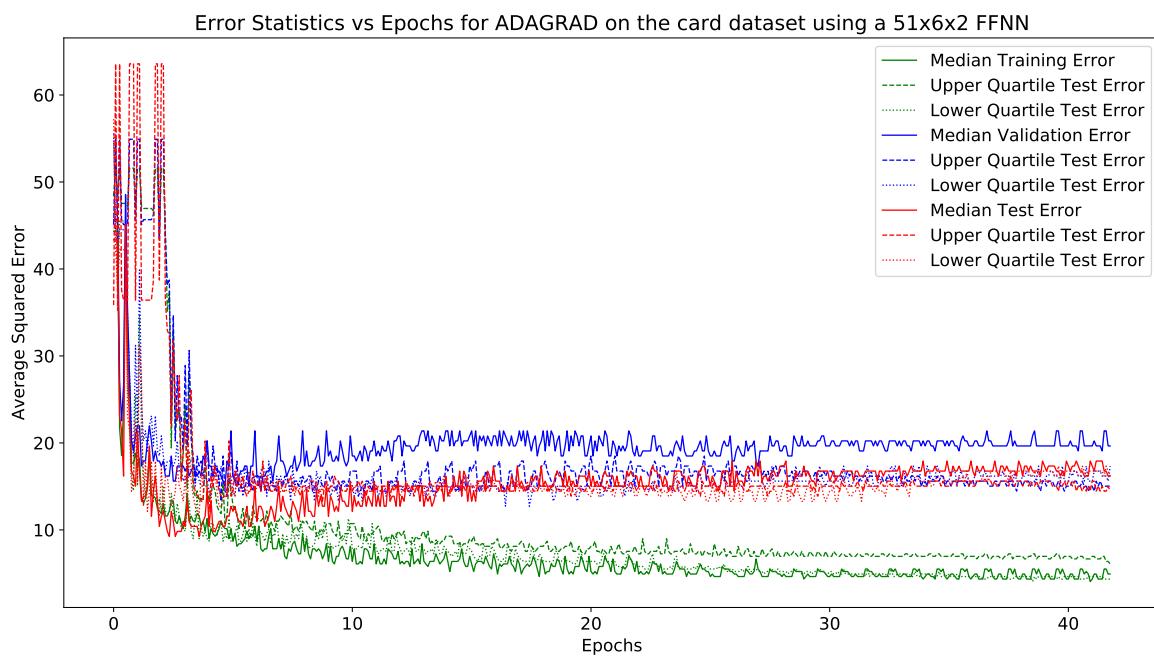
This section displays the results obtained using the card algorithm.

#### 51 × 6 × 2 Architecture:

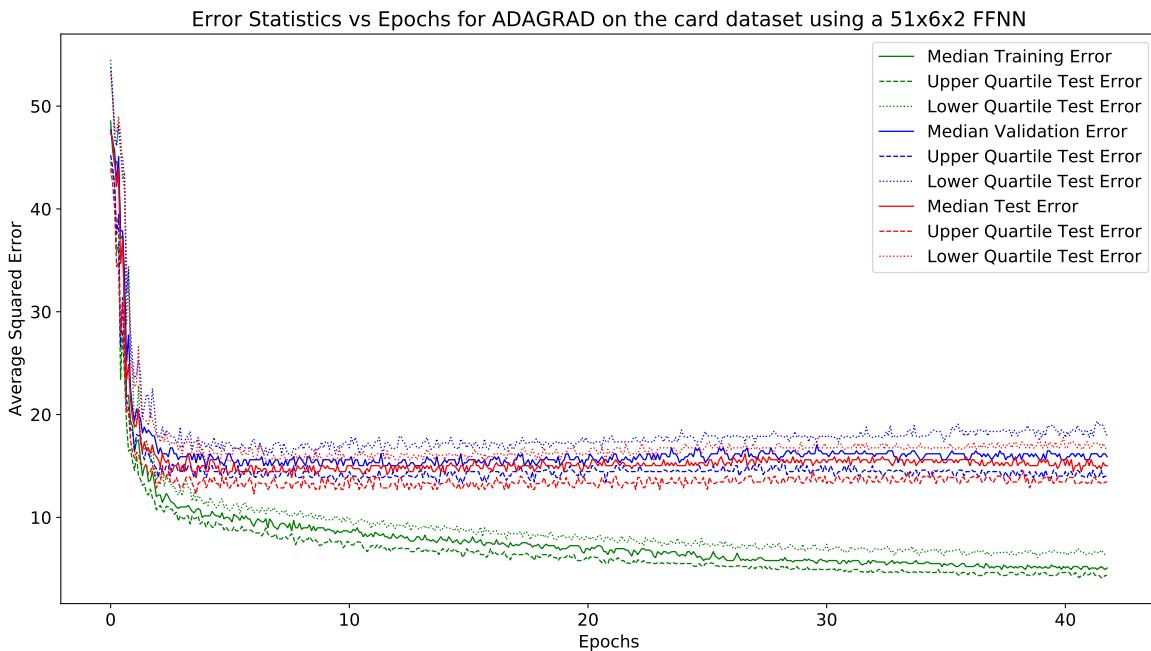
Fig. 152 shows the average and standard deviation of the test, training and validation errors. Fig. 153 shows the median and quartiles of the test, training and validation errors based on the best epoch. Finally, Fig. 154 shows the median and quartiles of the test, training and validation errors per mini-batch.



**Figure 152.** Graph of mean and standard deviation of errors vs epochs



**Figure 153.** Graph of test error vs epochs for the gradient based algorithms



**Figure 154.** Graph of test error vs epochs for the gradient based algorithms

#### 16.2.4 Backprop with Momentum

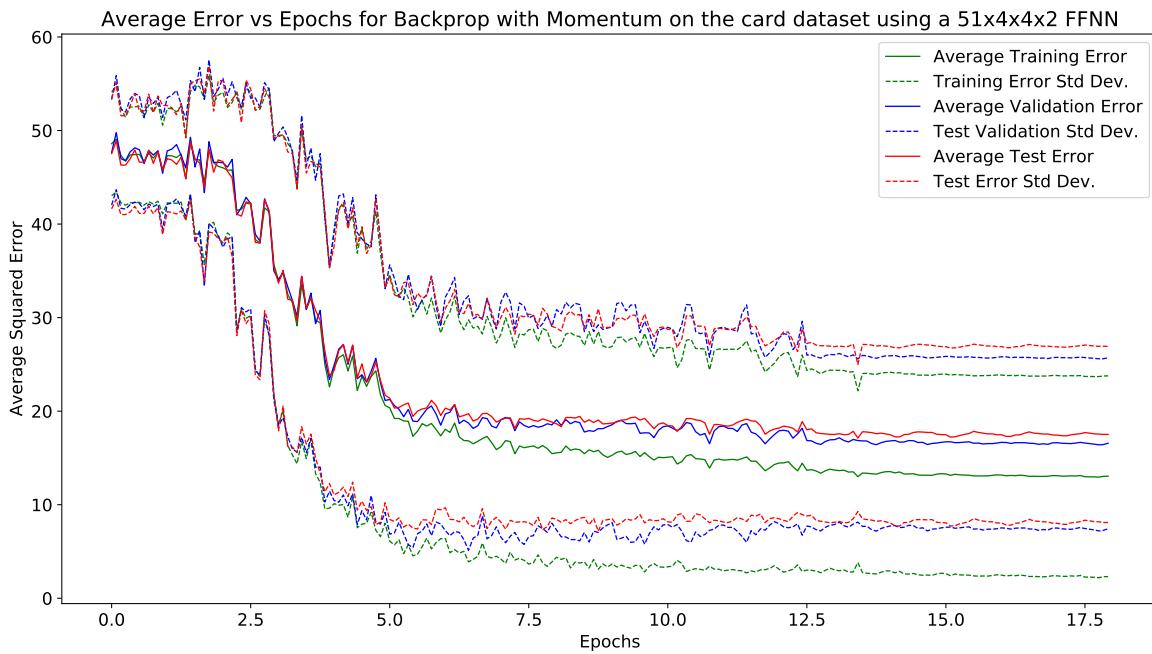
This section displays the results obtained using the Backprop with Momentum algorithm.

#### 16.2.5 card

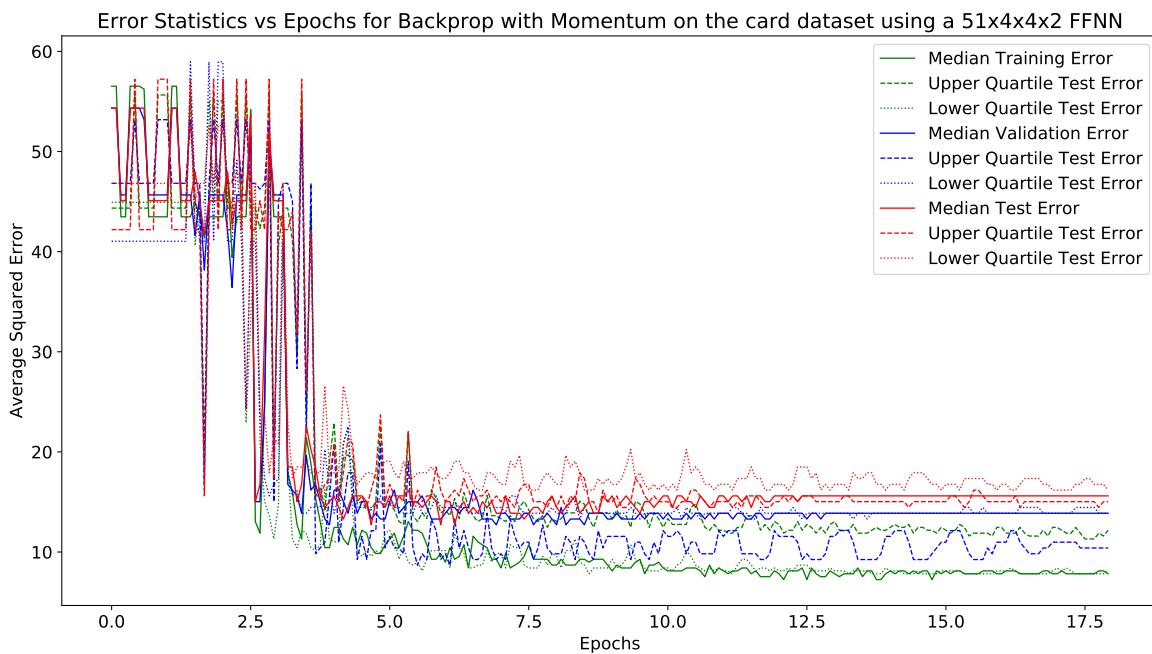
This section displays the results obtained using the card algorithm.

##### 51 × 4 × 4 × 2 Architecture:

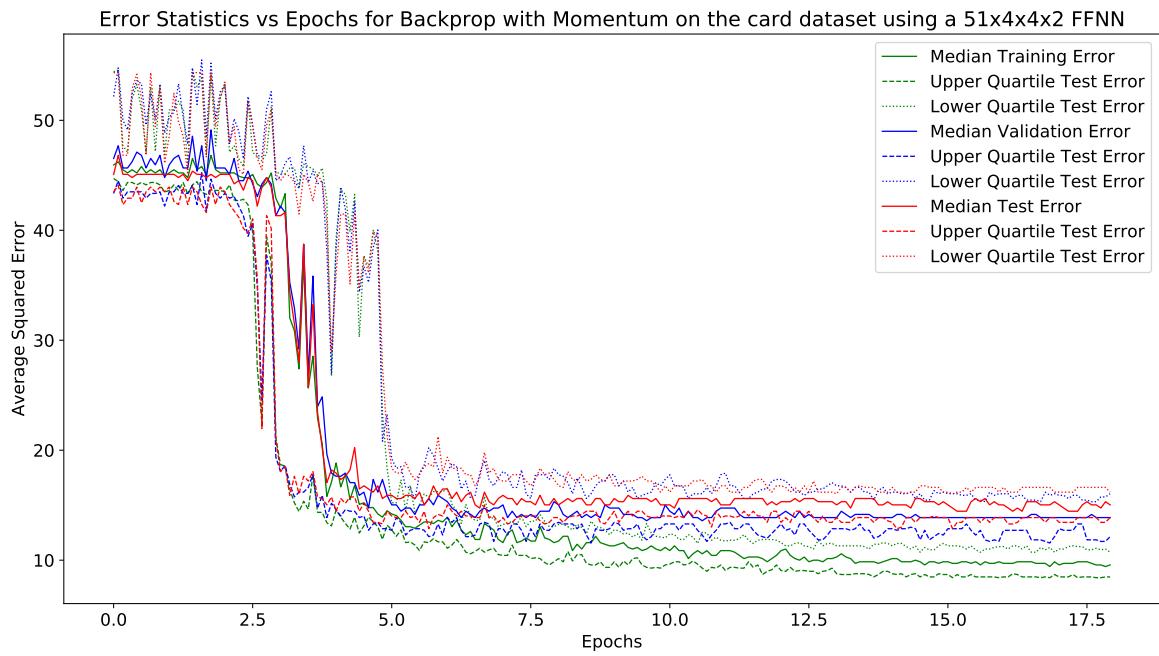
Fig. 155 shows the average and standard deviation of the test, training and validation errors. Fig. 156 shows the median and quartiles of the test, training and validation errors based on the best epoch. Finally, Fig. 157 shows the median and quartiles of the test, training and validation errors per mini-batch.



**Figure 155.** Graph of mean and standard deviation of errors vs epochs



**Figure 156.** Graph of test error vs epochs for the gradient based algorithms



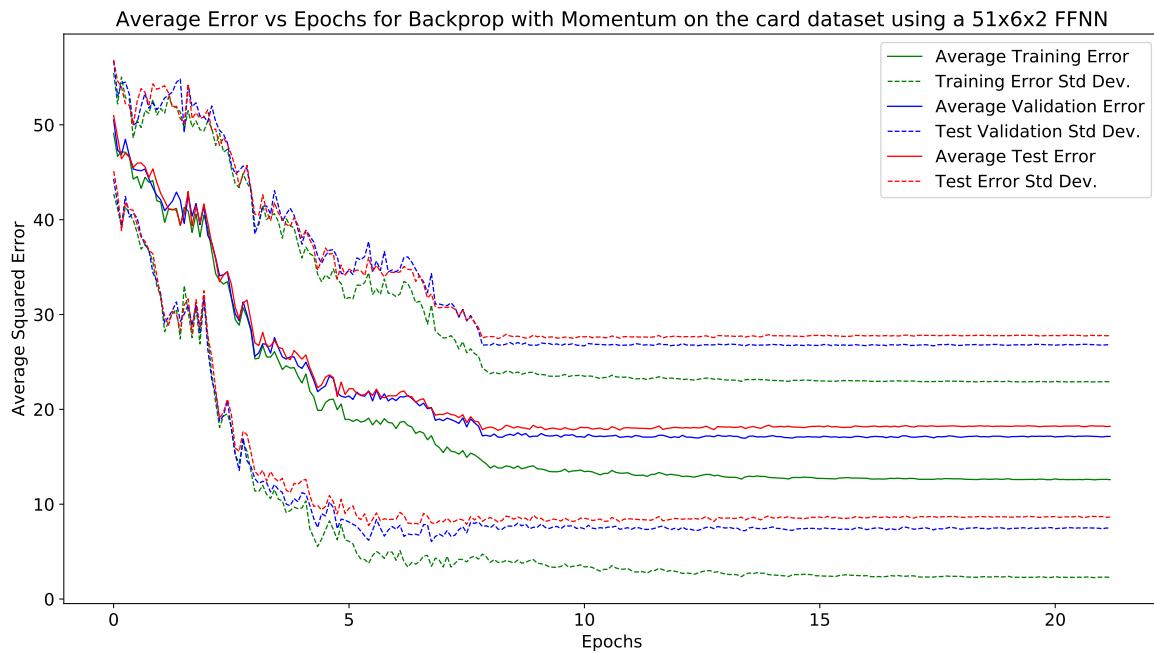
**Figure 157.** Graph of test error vs epochs for the gradient based algorithms

### 16.2.6 card

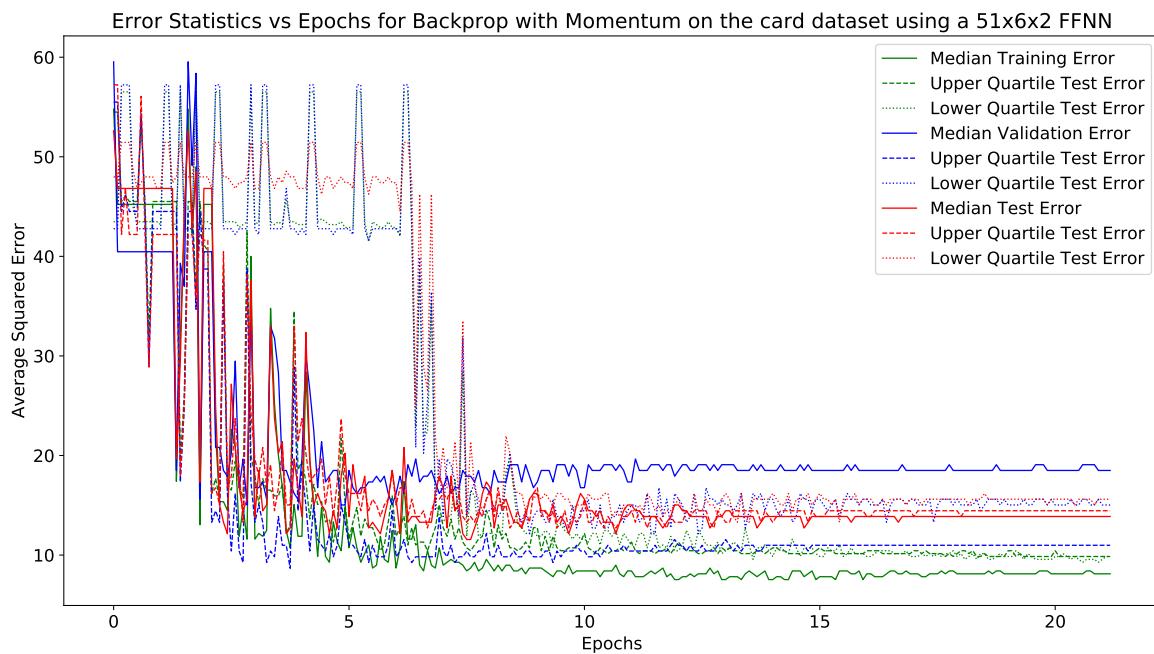
This section displays the results obtained using the card algorithm.

#### 51 × 6 × 2 Architecture:

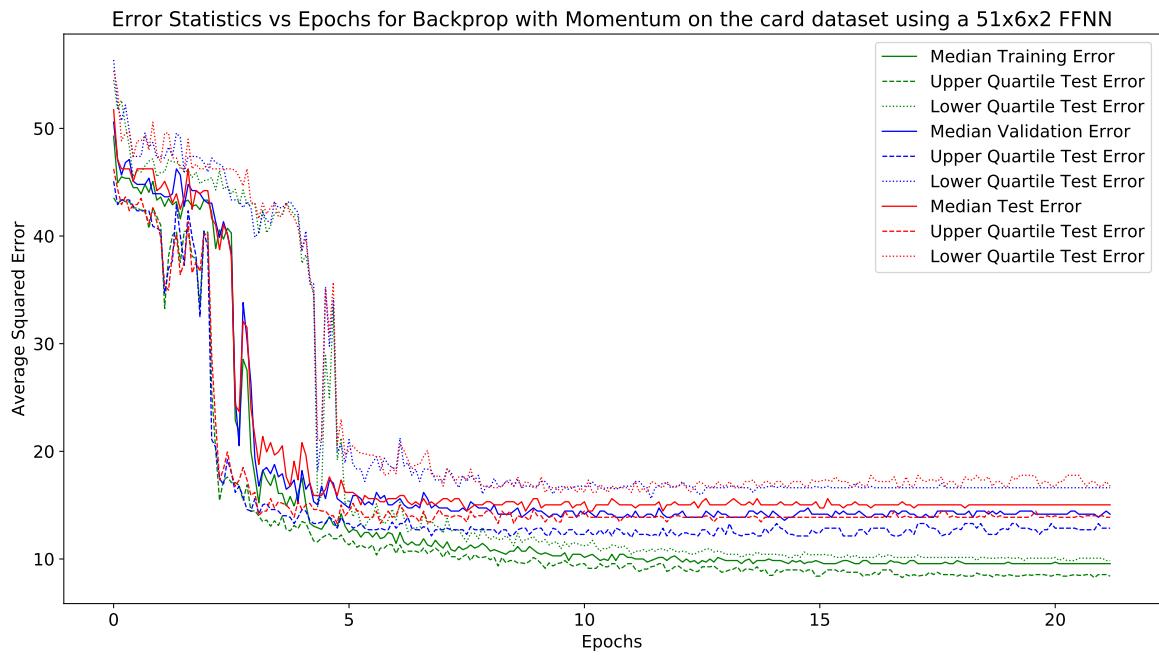
Fig. 158 shows the average and standard deviation of the test, training and validation errors. Fig. 159 shows the median and quartiles of the test, training and validation errors based on the best epoch. Finally, Fig. 160 shows the median and quartiles of the test, training and validation errors per mini-batch.



**Figure 158.** Graph of mean and standard deviation of errors vs epochs



**Figure 159.** Graph of test error vs epochs for the gradient based algorithms



**Figure 160.** Graph of test error vs epochs for the gradient based algorithms

### 16.2.7 back-propogation

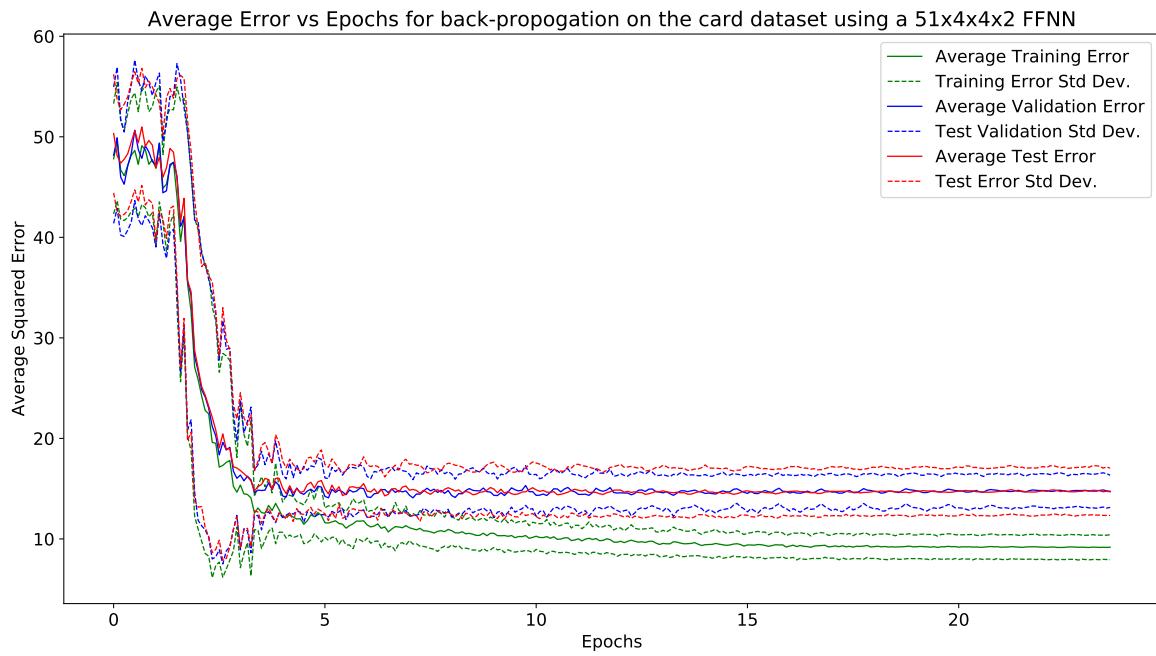
This section displays the results obtained using the back-propogation algorithm.

### 16.2.8 card

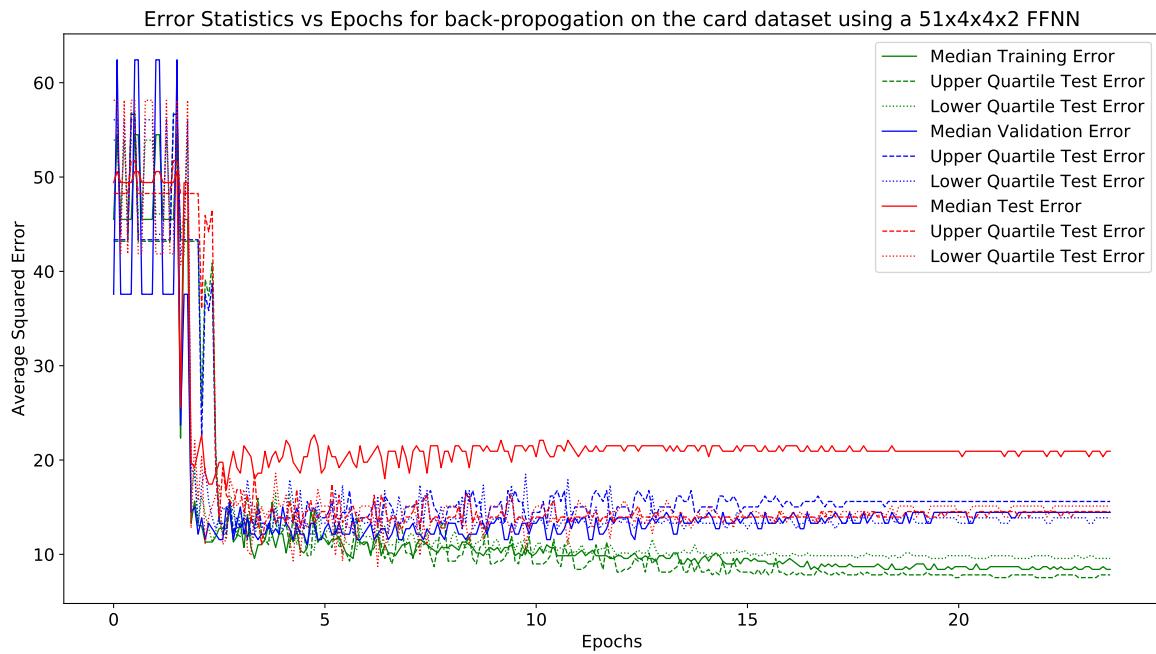
This section displays the results obtained using the card algorithm.

#### 51 × 4 × 4 × 2 Architecture:

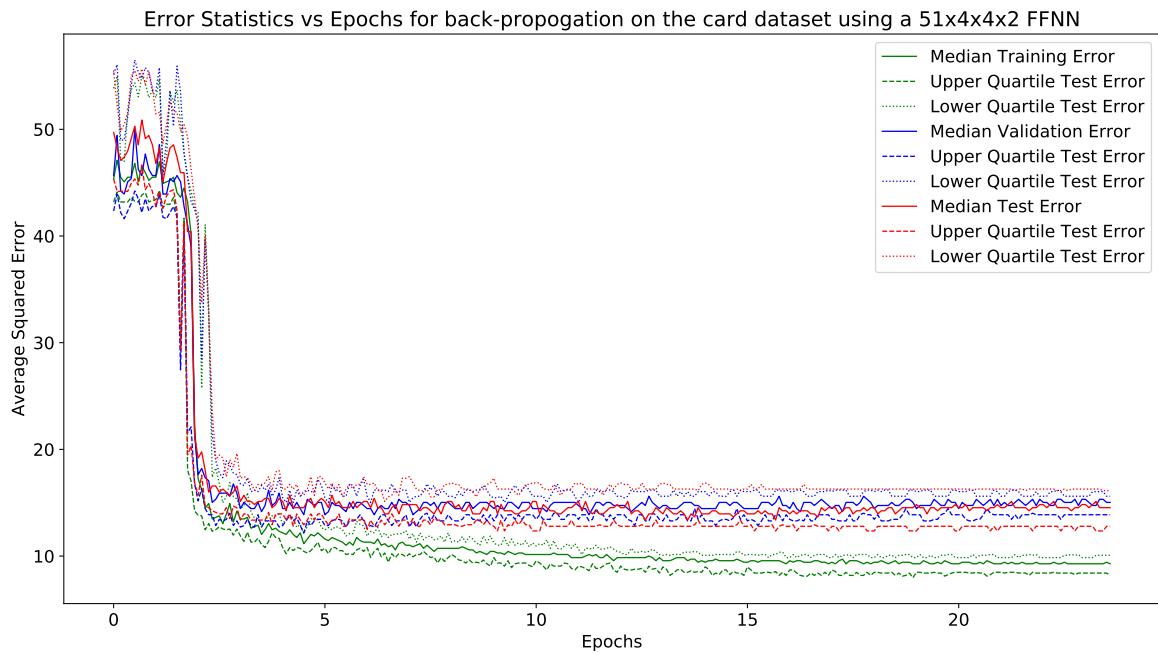
Fig. 161 shows the average and standard deviation of the test, training and validation errors. Fig. 162 shows the median and quartiles of the test, training and validation errors based on the best epoch. Finally, Fig. 163 shows the median and quartiles of the test, training and validation errors per mini-batch.



**Figure 161.** Graph of mean and standard deviation of errors vs epochs



**Figure 162.** Graph of test error vs epochs for the gradient based algorithms



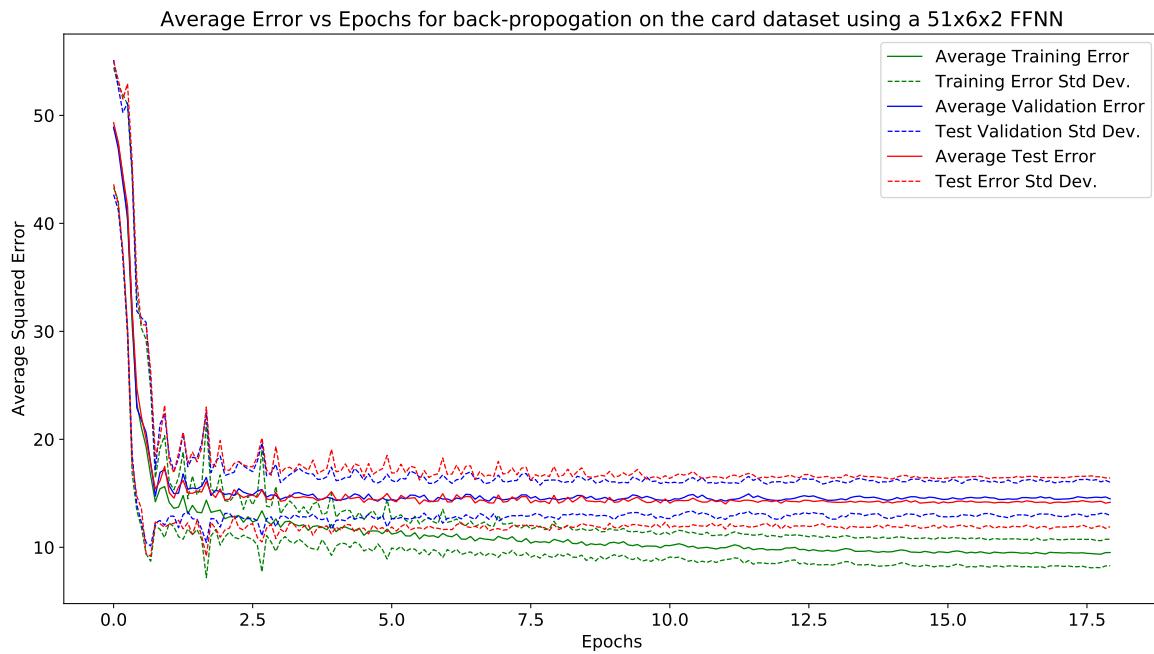
**Figure 163.** Graph of test error vs epochs for the gradient based algorithms

### 16.2.9 card

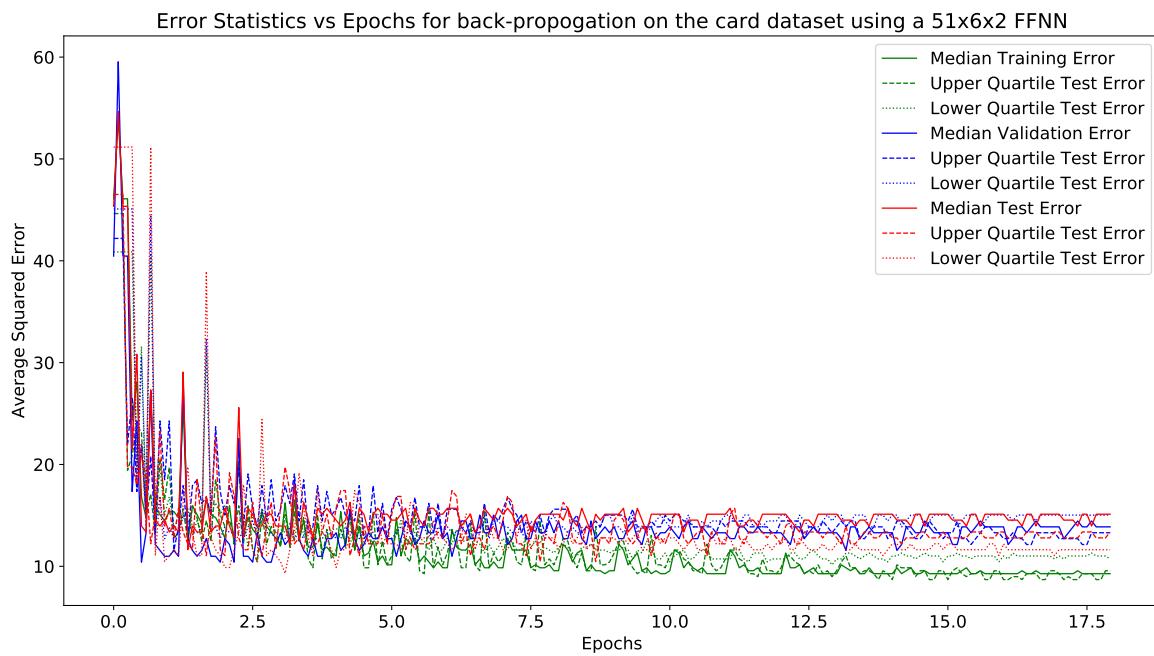
This section displays the results obtained using the card algorithm.

#### 51 × 6 × 2 Architecture:

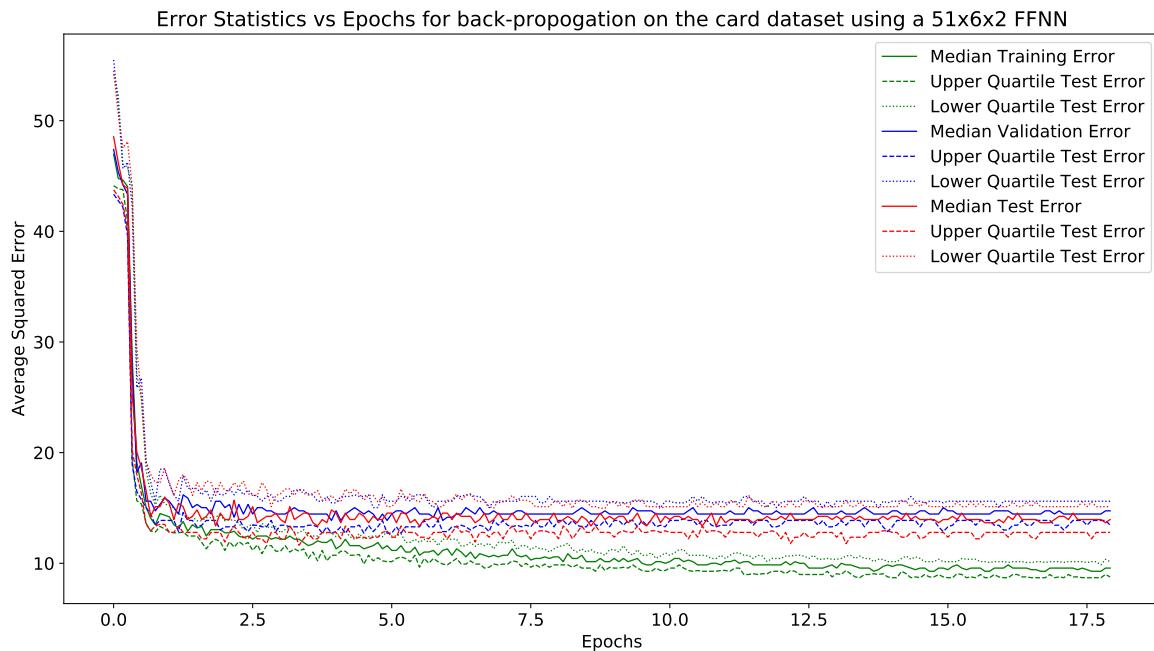
Fig. 164 shows the average and standard deviation of the test, training and validation errors. Fig. 165 shows the median and quartiles of the test, training and validation errors based on the best epoch. Finally, Fig. 166 shows the median and quartiles of the test, training and validation errors per mini-batch.



**Figure 164.** Graph of mean and standard deviation of errors vs epochs



**Figure 165.** Graph of test error vs epochs for the gradient based algorithms



**Figure 166.** Graph of test error vs epochs for the gradient based algorithms

### 16.2.10 GA1

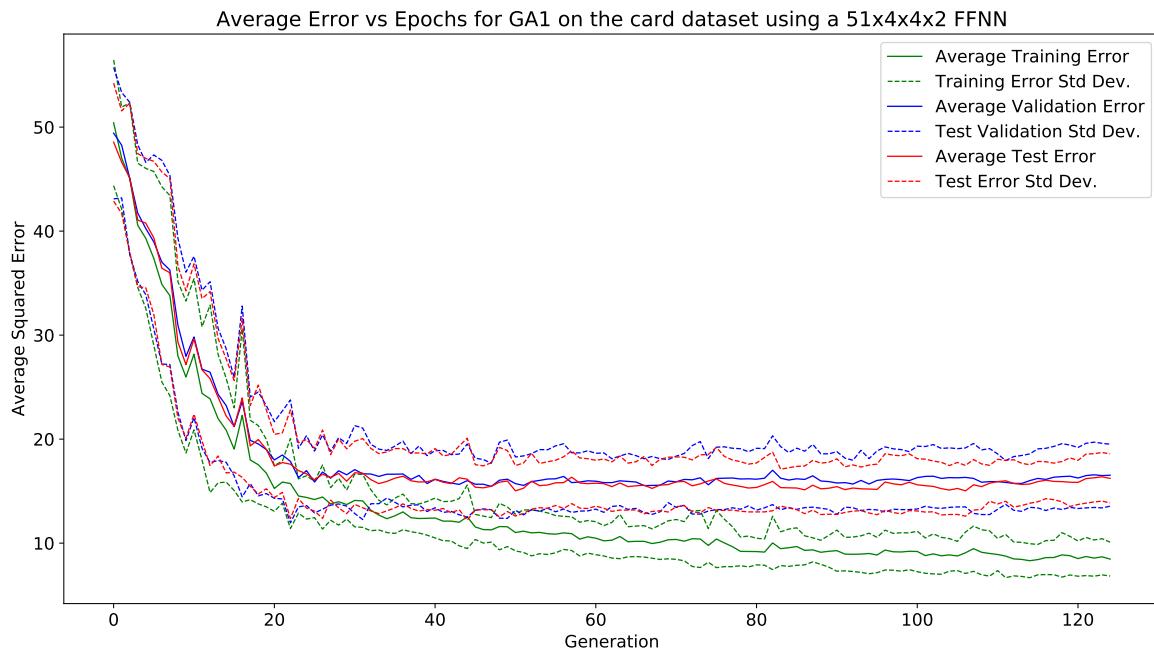
This section displays the results obtained using the GA1 algorithm.

### 16.2.11 card

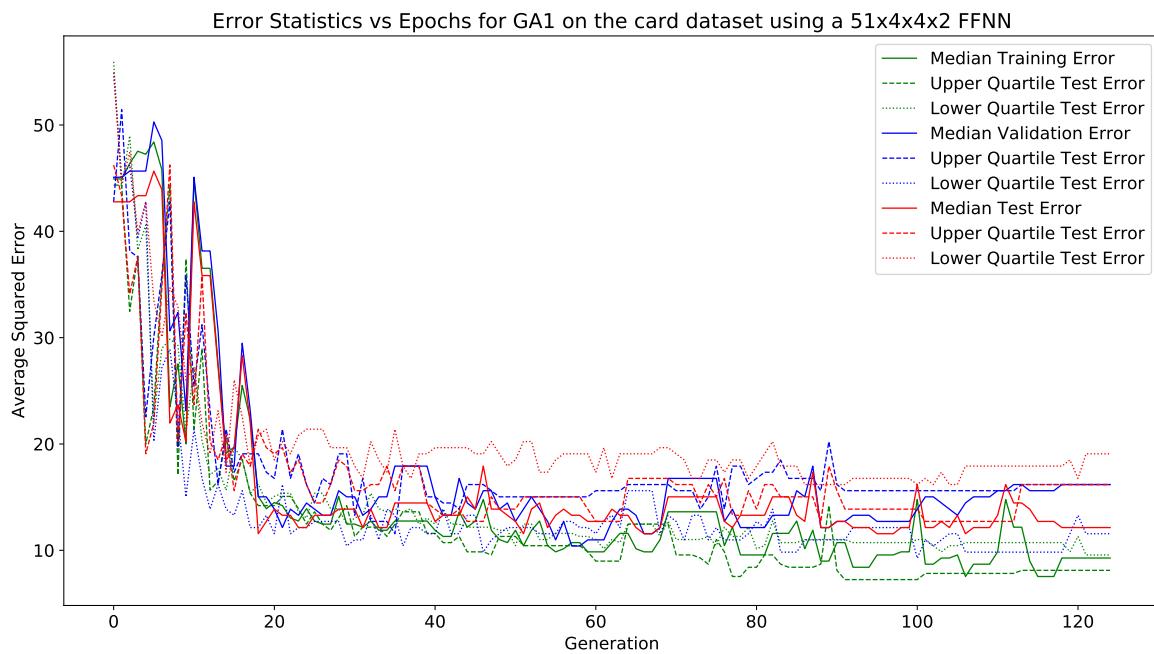
This section displays the results obtained using the card algorithm.

#### 51 × 4 × 4 × 2 Architecture:

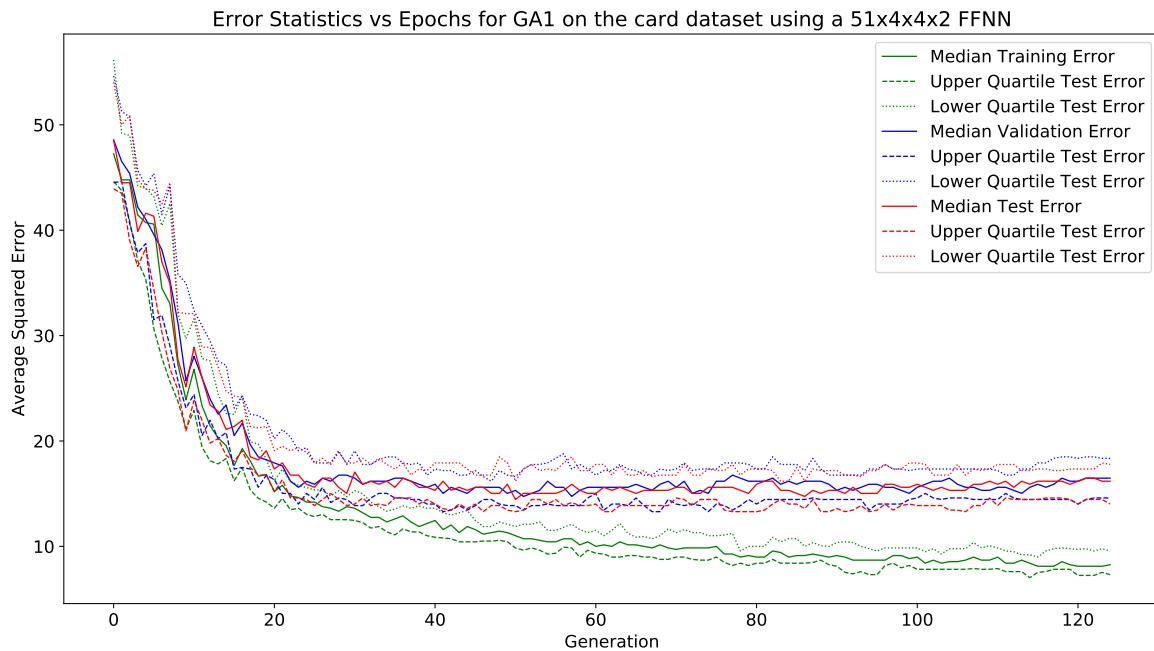
Fig. 167 shows the average and standard deviation of the test, training and validation errors. Fig. 168 shows the median and quartiles of the test, training and validation errors based on the best epoch. Finally, Fig. 169 shows the median and quartiles of the test, training and validation errors per mini-batch.



**Figure 167.** Graph of mean and standard deviation of errors vs epochs



**Figure 168.** Graph of test error vs epochs for the gradient based algorithms



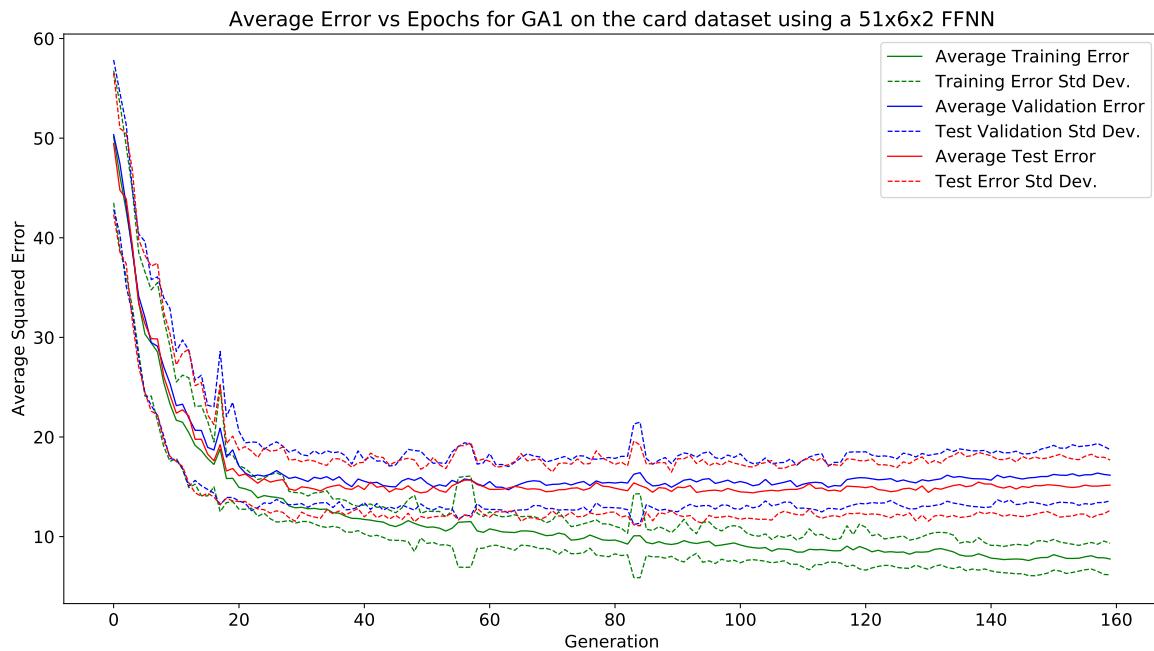
**Figure 169.** Graph of test error vs epochs for the gradient based algorithms

### 16.2.12 card

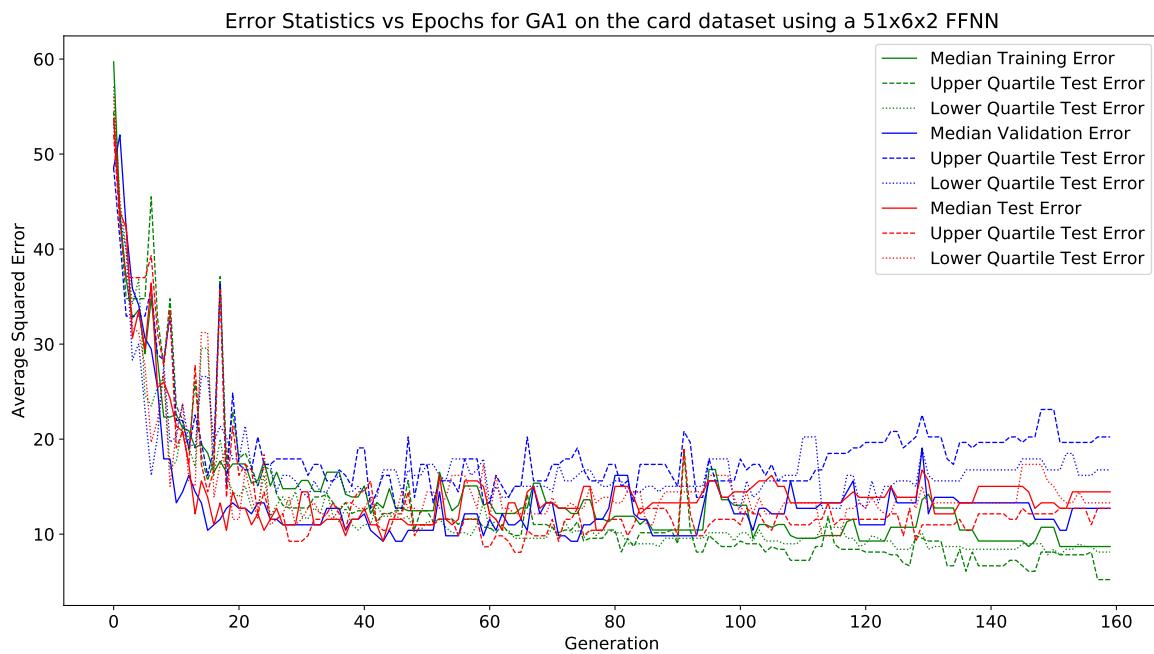
This section displays the results obtained using the card algorithm.

#### 51 × 6 × 2 Architecture:

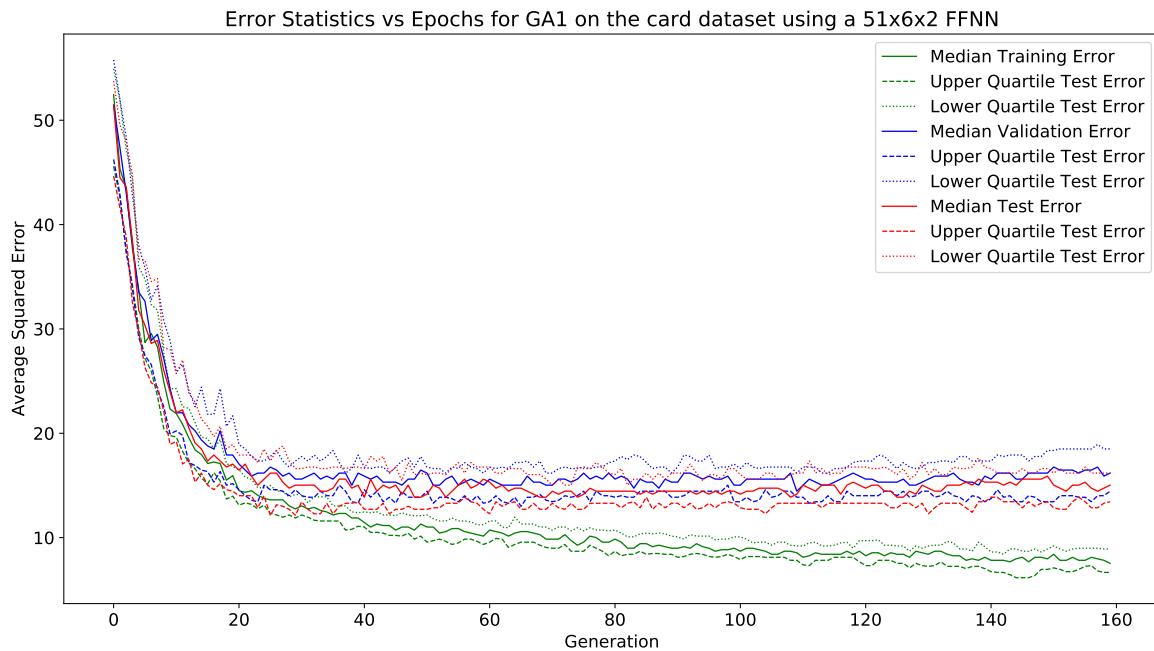
Fig. 170 shows the average and standard deviation of the test, training and validation errors. Fig. 171 shows the median and quartiles of the test, training and validation errors based on the best epoch. Finally, Fig. 172 shows the median and quartiles of the test, training and validation errors per mini-batch.



**Figure 170.** Graph of mean and standard deviation of errors vs epochs



**Figure 171.** Graph of test error vs epochs for the gradient based algorithms



**Figure 172.** Graph of test error vs epochs for the gradient based algorithms

### 16.2.13 GA2

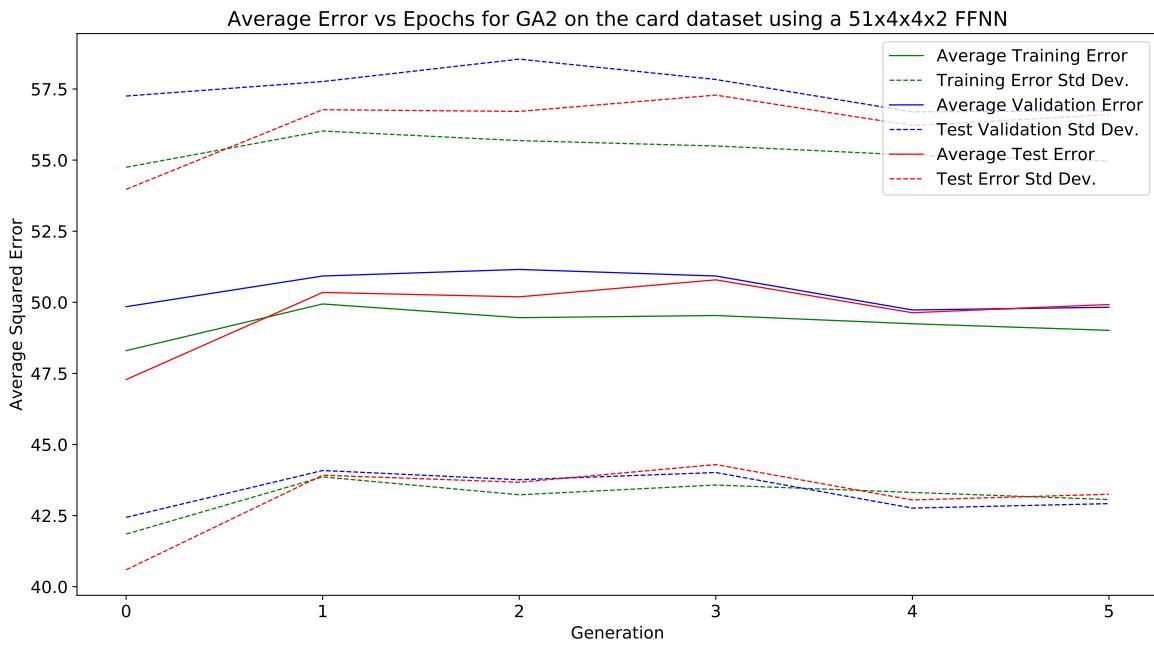
This section displays the results obtained using the GA2 algorithm.

### 16.2.14 card

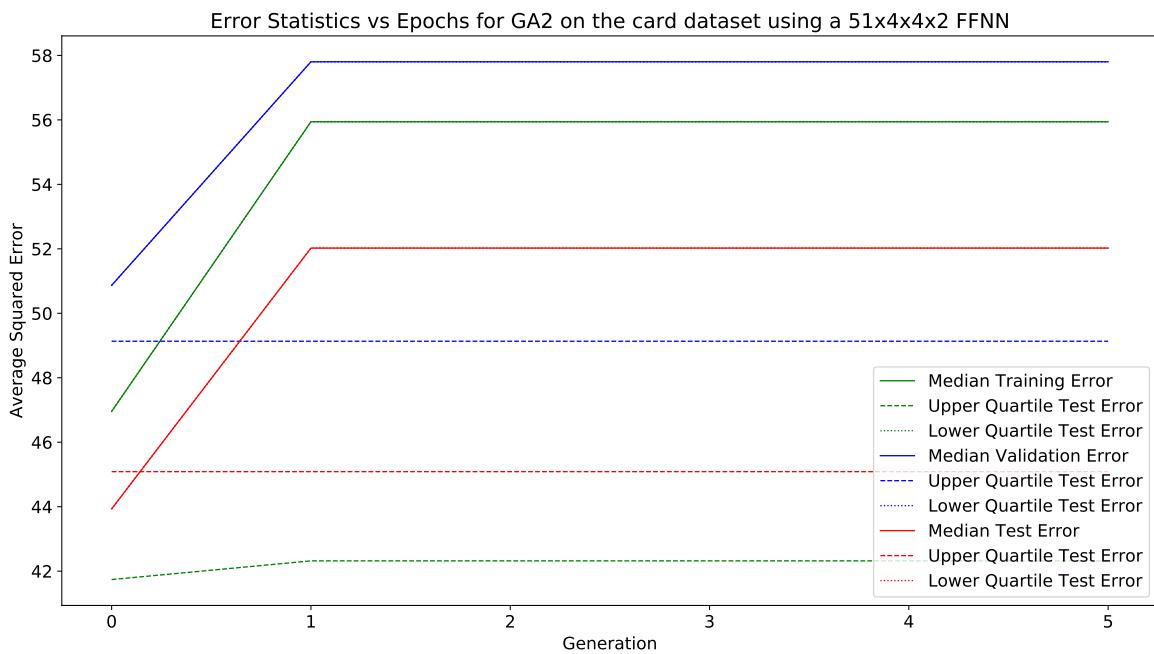
This section displays the results obtained using the card algorithm.

#### 51 × 4 × 4 × 2 Architecture:

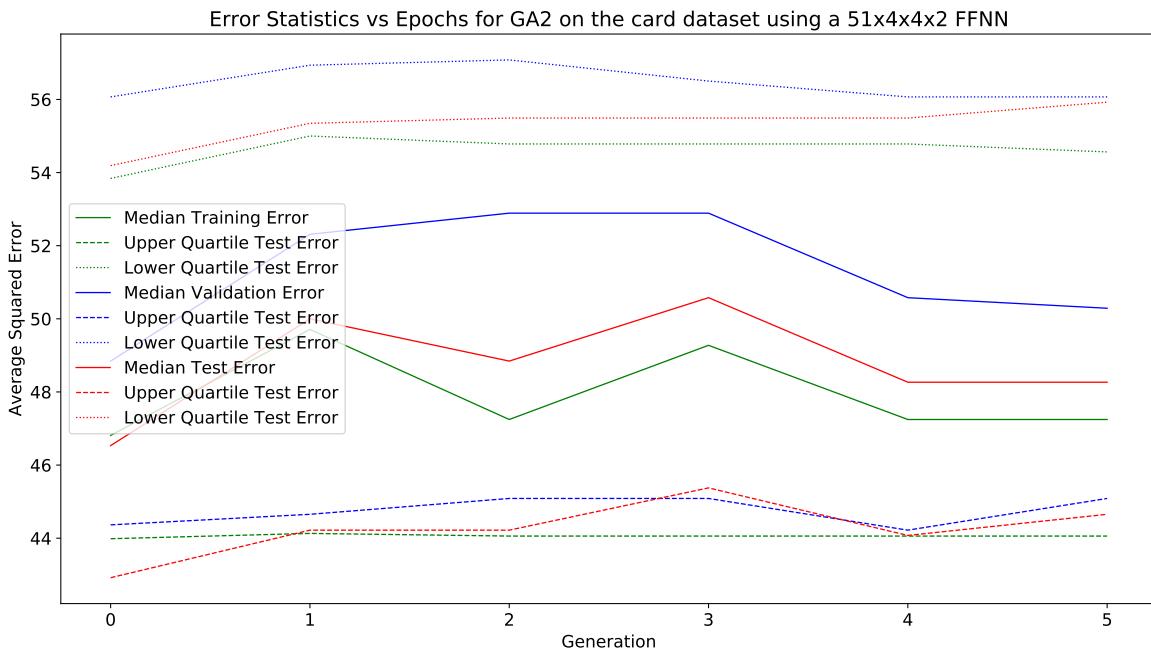
Fig. 173 shows the average and standard deviation of the test, training and validation errors. Fig. 174 shows the median and quartiles of the test, training and validation errors based on the best epoch. Finally, Fig. 175 shows the median and quartiles of the test, training and validation errors per mini-batch.



**Figure 173.** Graph of mean and standard deviation of errors vs epochs



**Figure 174.** Graph of test error vs epochs for the gradient based algorithms



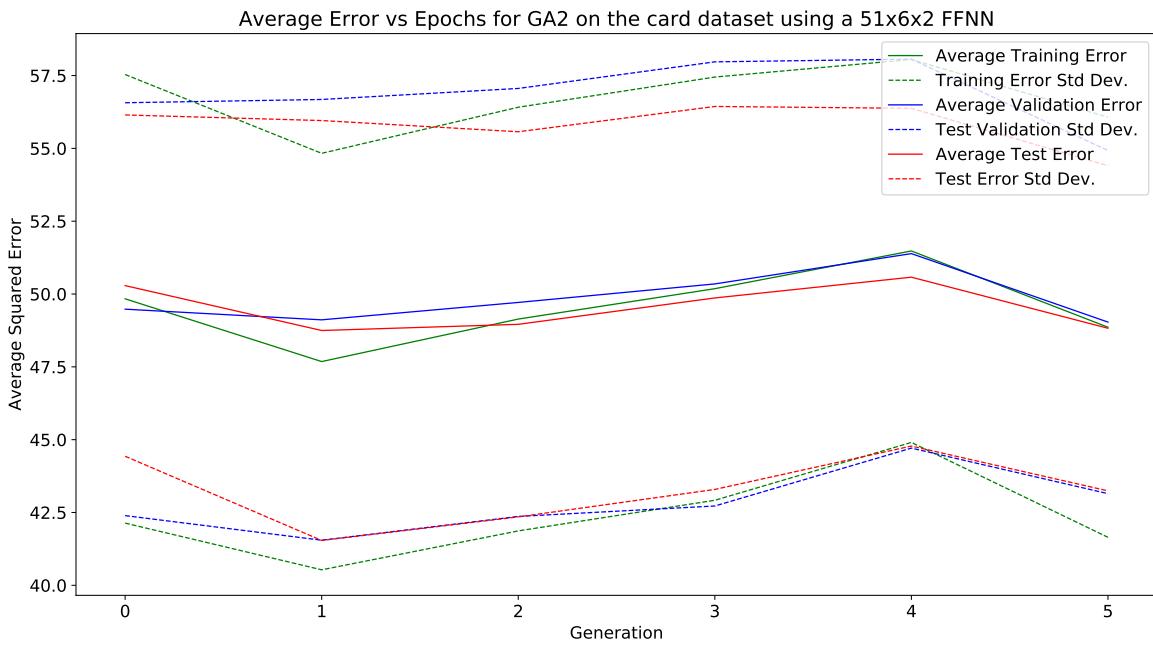
**Figure 175.** Graph of test error vs epochs for the gradient based algorithms

### 16.2.15 card

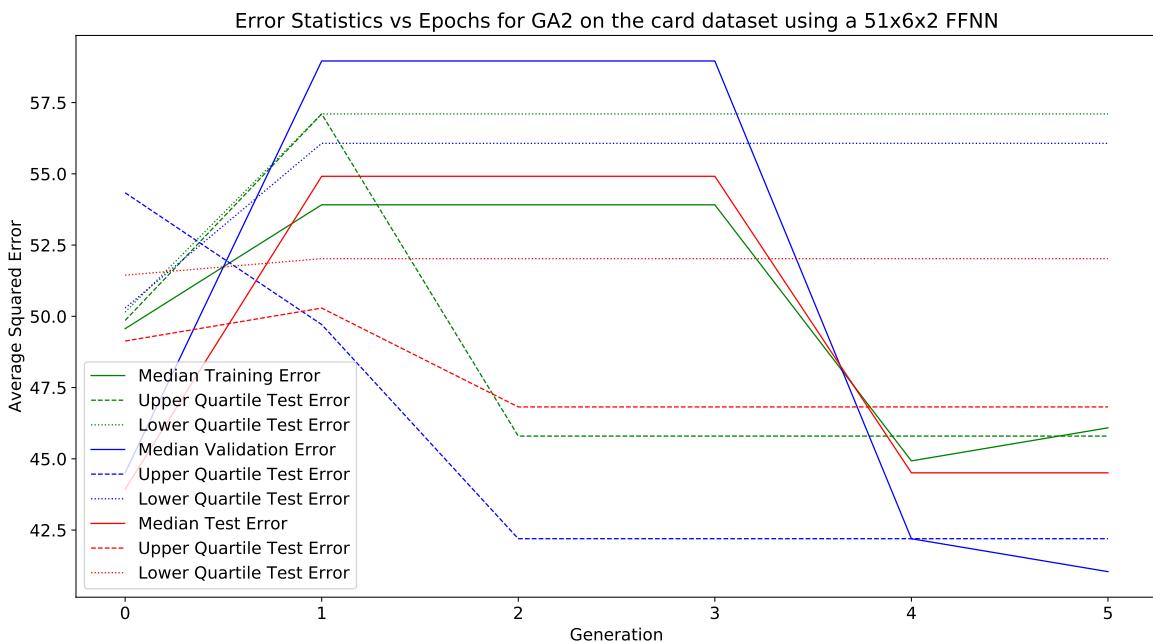
This section displays the results obtained using the card algorithm.

#### 51 × 6 × 2 Architecture:

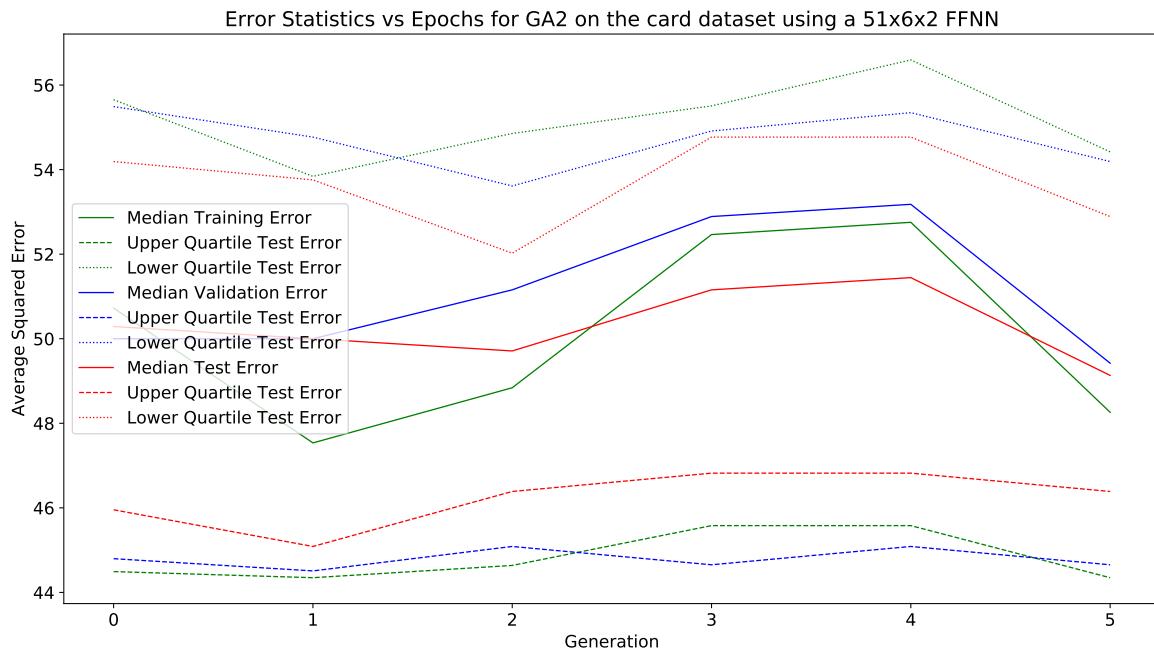
Fig. 176 shows the average and standard deviation of the test, training and validation errors. Fig. 177 shows the median and quartiles of the test, training and validation errors based on the best epoch. Finally, Fig. 178 shows the median and quartiles of the test, training and validation errors per mini-batch.



**Figure 176.** Graph of mean and standard deviation of errors vs epochs



**Figure 177.** Graph of test error vs epochs for the gradient based algorithms



**Figure 178.** Graph of test error vs epochs for the gradient based algorithms

### 16.2.16 GA3

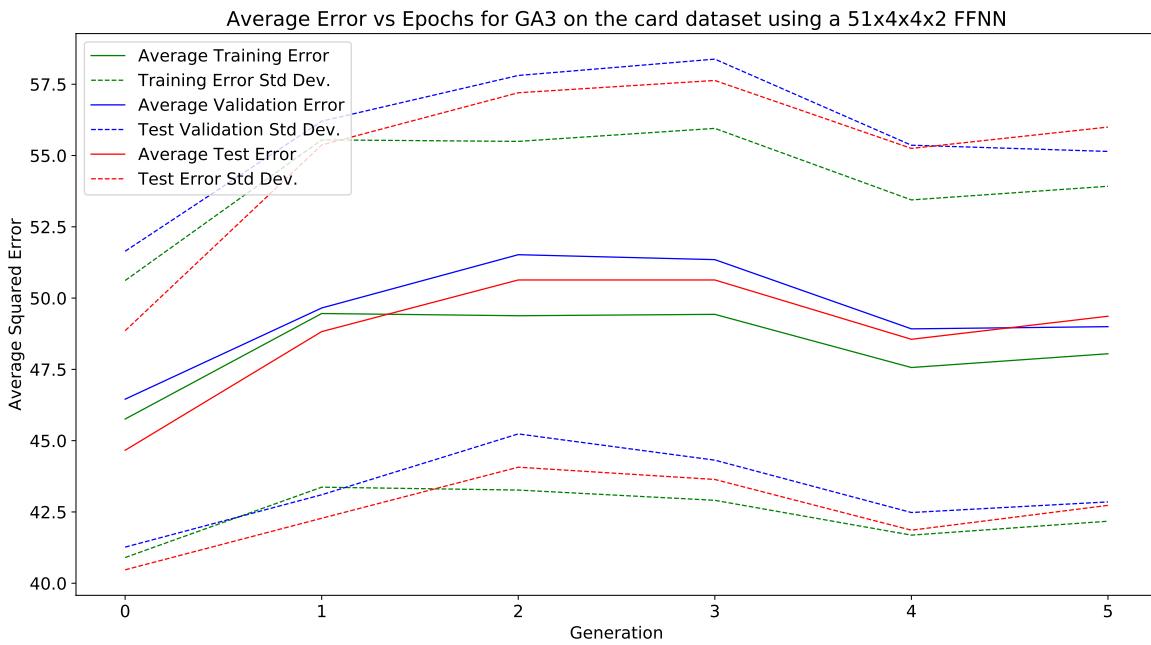
This section displays the results obtained using the GA3 algorithm.

### 16.2.17 card

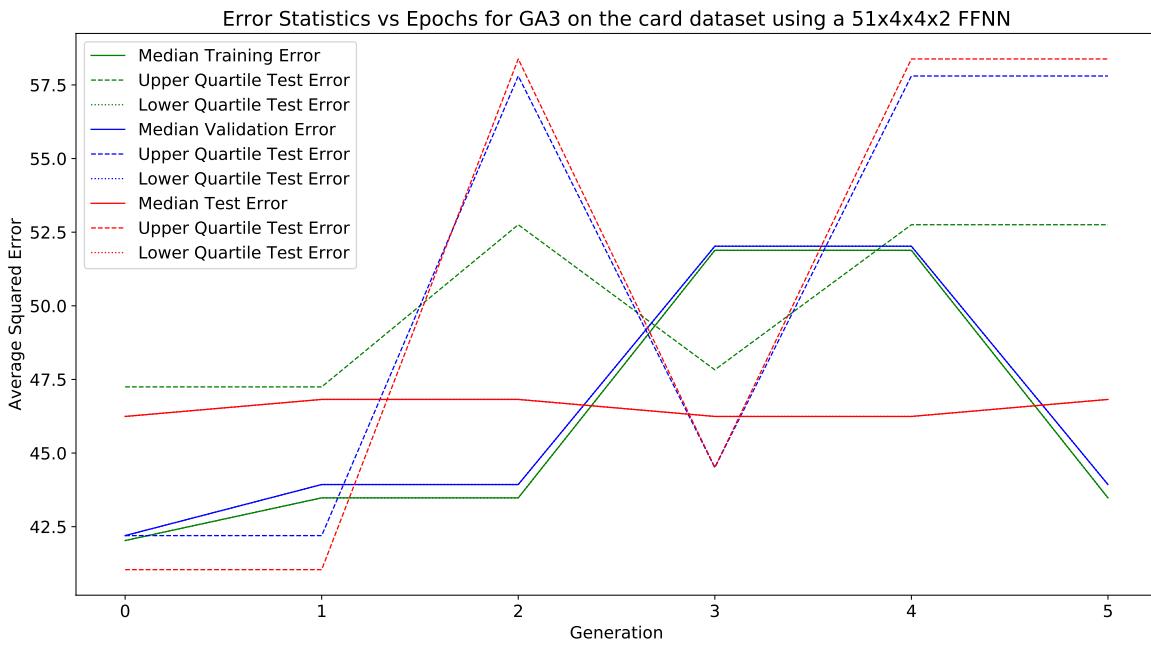
This section displays the results obtained using the card algorithm.

#### 51 × 4 × 4 × 2 Architecture:

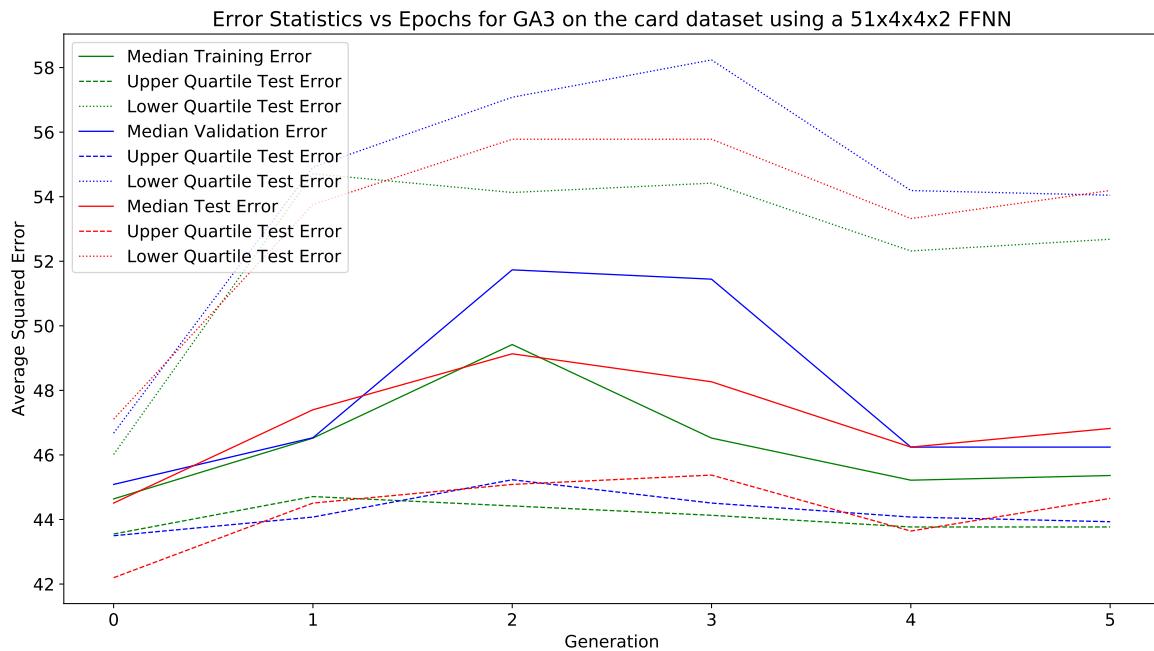
Fig. 179 shows the average and standard deviation of the test, training and validation errors. Fig. 180 shows the median and quartiles of the test, training and validation errors based on the best epoch. Finally, Fig. 181 shows the median and quartiles of the test, training and validation errors per mini-batch.



**Figure 179.** Graph of mean and standard deviation of errors vs epochs



**Figure 180.** Graph of test error vs epochs for the gradient based algorithms



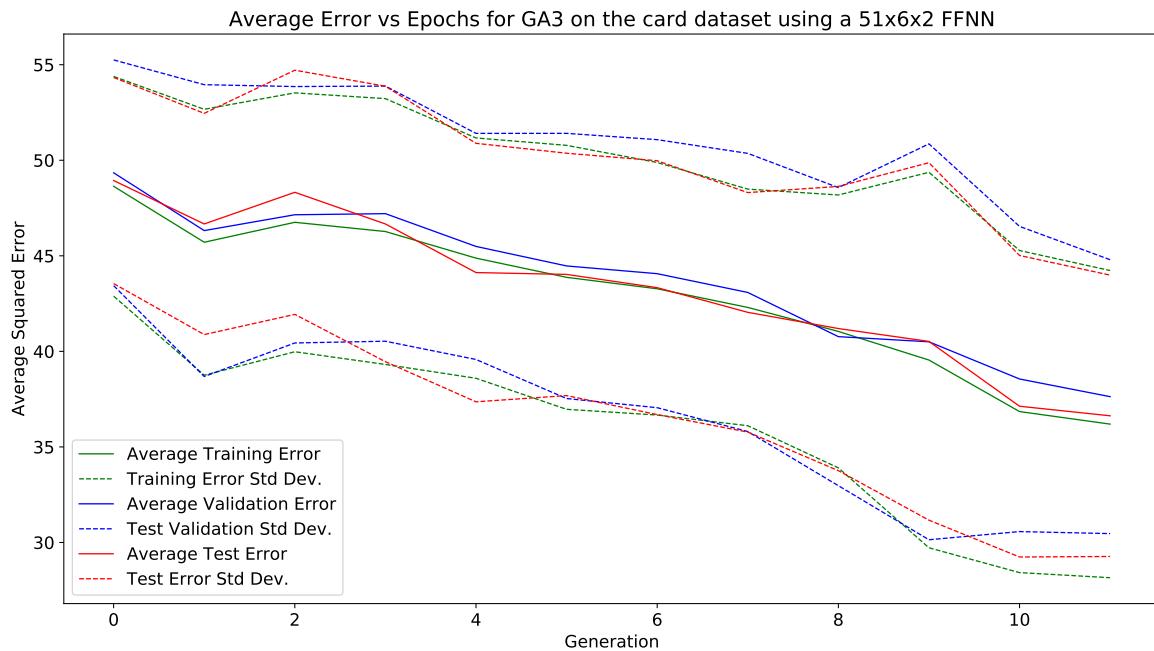
**Figure 181.** Graph of test error vs epochs for the gradient based algorithms

### 16.2.18 card

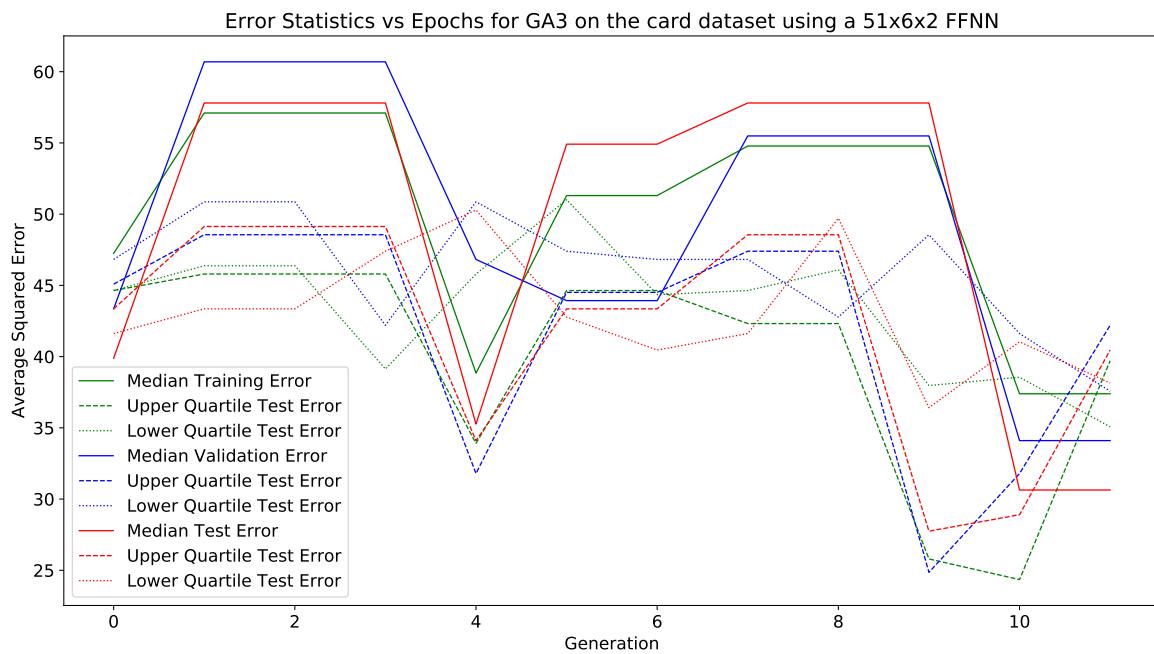
This section displays the results obtained using the card algorithm.

#### 51 × 6 × 2 Architecture:

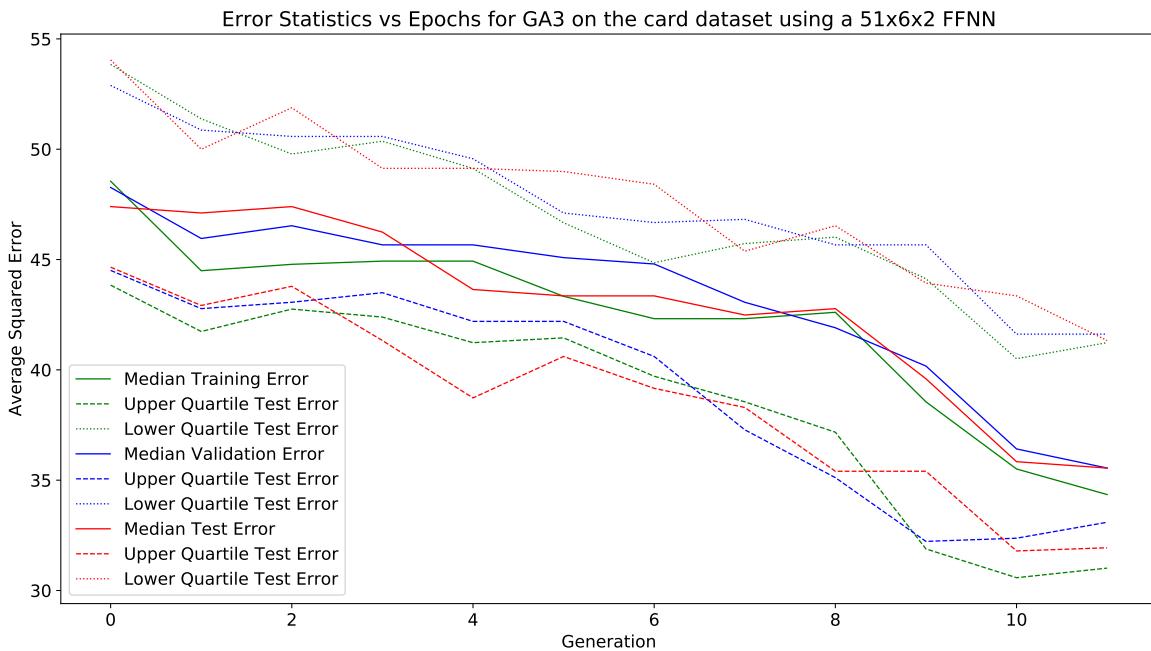
Fig. 182 shows the average and standard deviation of the test, training and validation errors. Fig. 183 shows the median and quartiles of the test, training and validation errors based on the best epoch. Finally, Fig. 184 shows the median and quartiles of the test, training and validation errors per mini-batch.



**Figure 182.** Graph of mean and standard deviation of errors vs epochs



**Figure 183.** Graph of test error vs epochs for the gradient based algorithms



**Figure 184.** Graph of test error vs epochs for the gradient based algorithms

### 16.2.19 iRPROP-

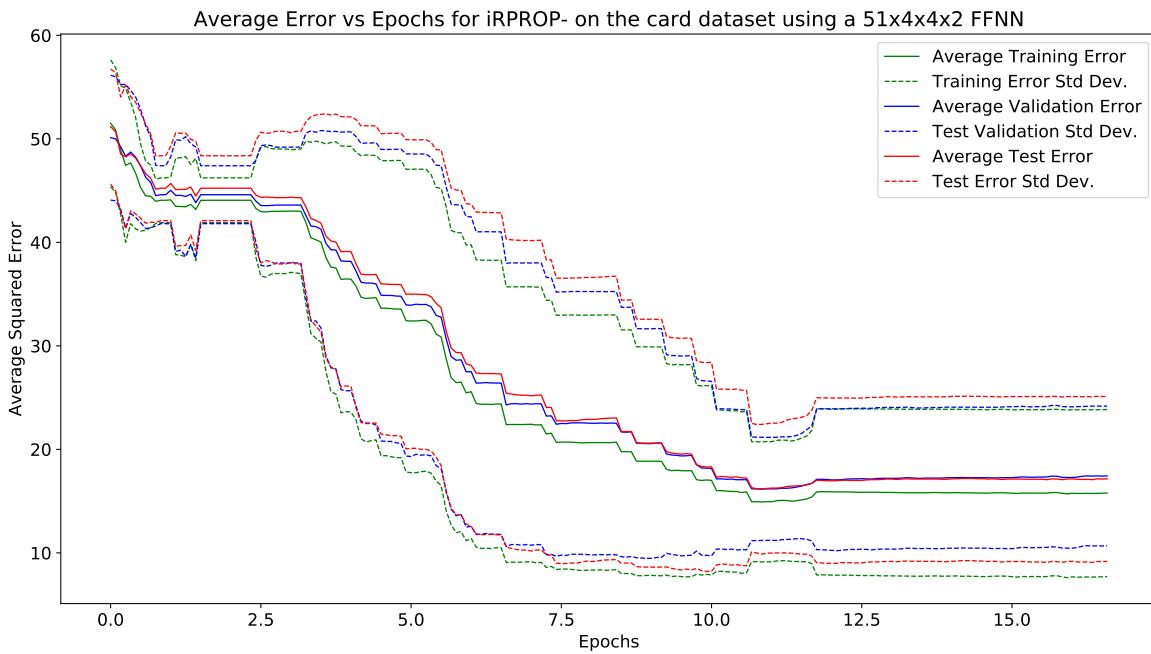
This section displays the results obtained using the iRPROP- algorithm.

### 16.2.20 card

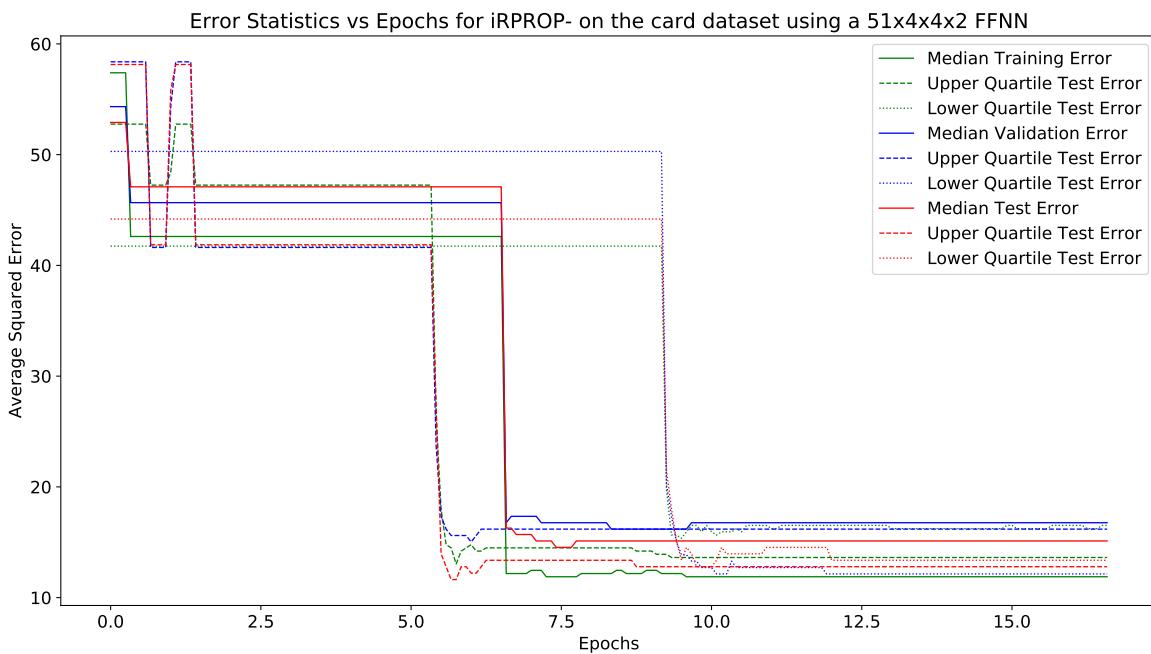
This section displays the results obtained using the card algorithm.

#### 51 × 4 × 4 × 2 Architecture:

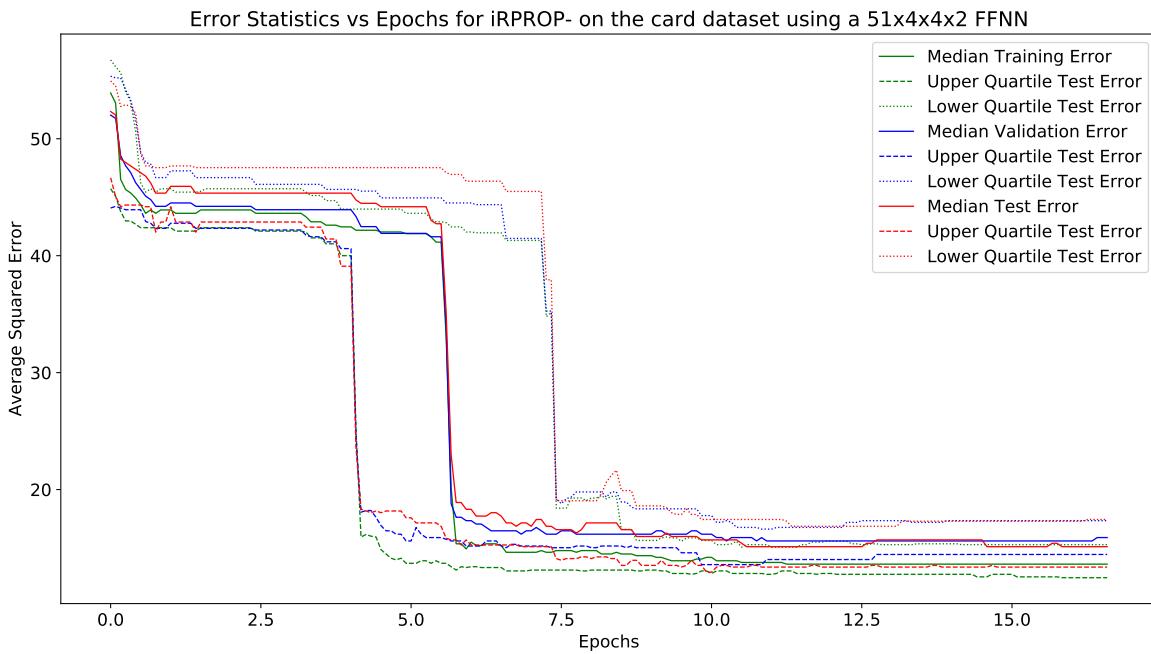
Fig. 185 shows the average and standard deviation of the test, training and validation errors. Fig. 186 shows the median and quartiles of the test, training and validation errors based on the best epoch. Finally, Fig. 187 shows the median and quartiles of the test, training and validation errors per mini-batch.



**Figure 185.** Graph of mean and standard deviation of errors vs epochs



**Figure 186.** Graph of test error vs epochs for the gradient based algorithms



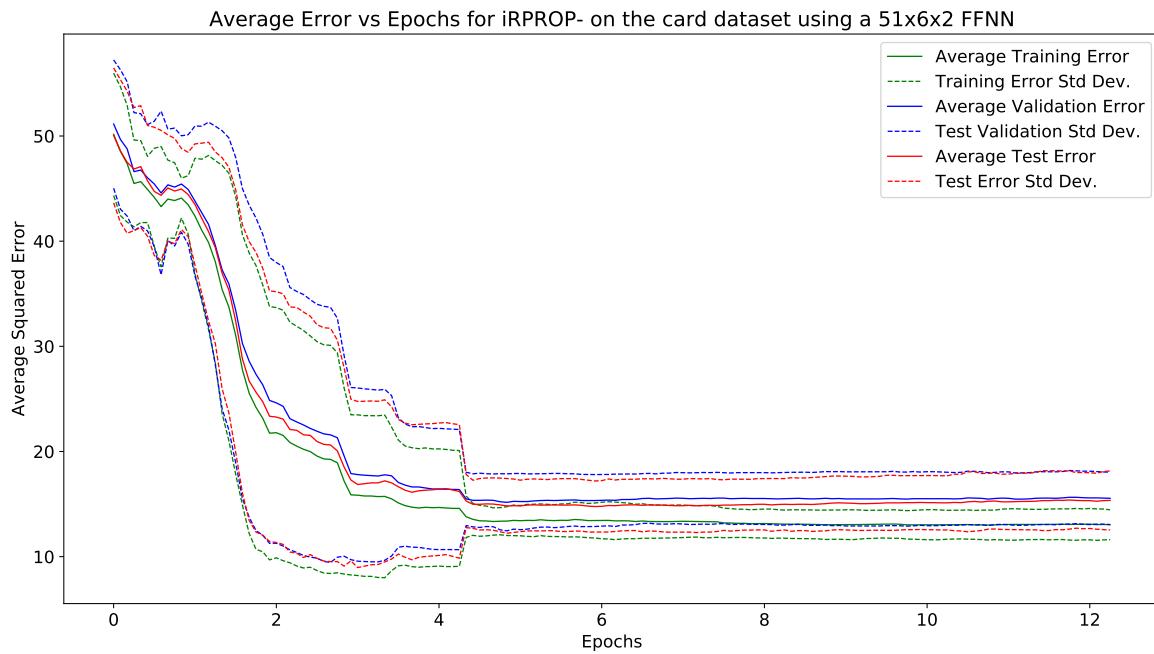
**Figure 187.** Graph of test error vs epochs for the gradient based algorithms

### 16.2.21 card

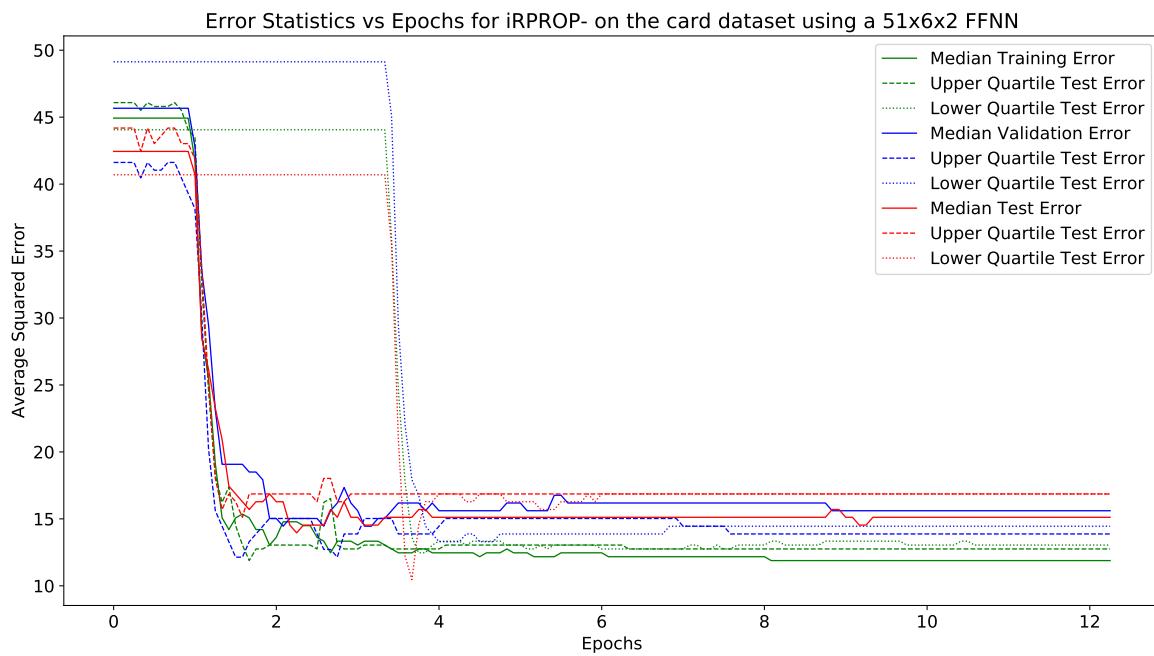
This section displays the results obtained using the card algorithm.

#### 51 × 6 × 2 Architecture:

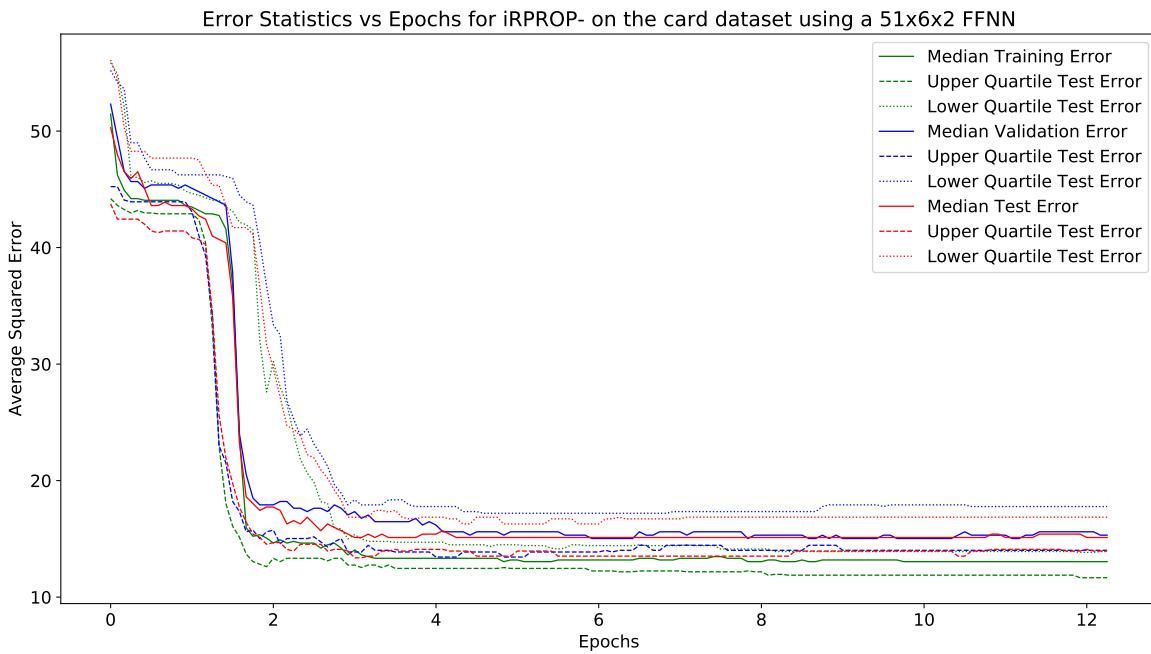
Fig. 188 shows the average and standard deviation of the test, training and validation errors. Fig. 189 shows the median and quartiles of the test, training and validation errors based on the best epoch. Finally, Fig. 190 shows the median and quartiles of the test, training and validation errors per mini-batch.



**Figure 188.** Graph of mean and standard deviation of errors vs epochs



**Figure 189.** Graph of test error vs epochs for the gradient based algorithms



**Figure 190.** Graph of test error vs epochs for the gradient based algorithms

### 16.2.22 *iRPROP+*

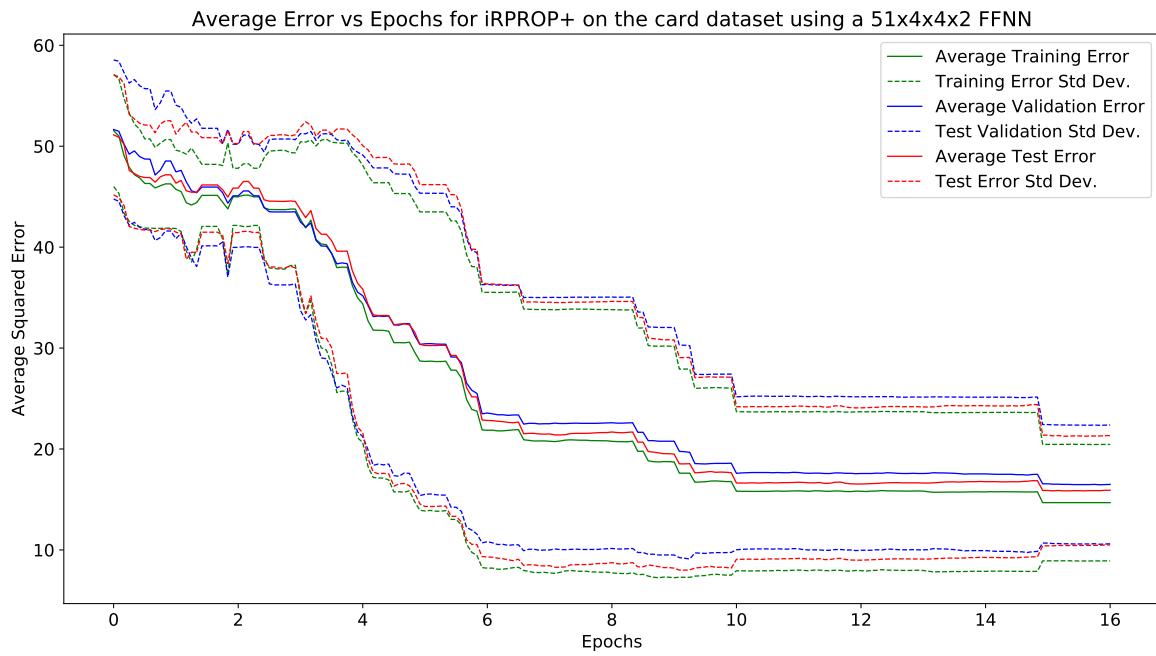
This section displays the results obtained using the iRPROP+ algorithm.

### 16.2.23 *card*

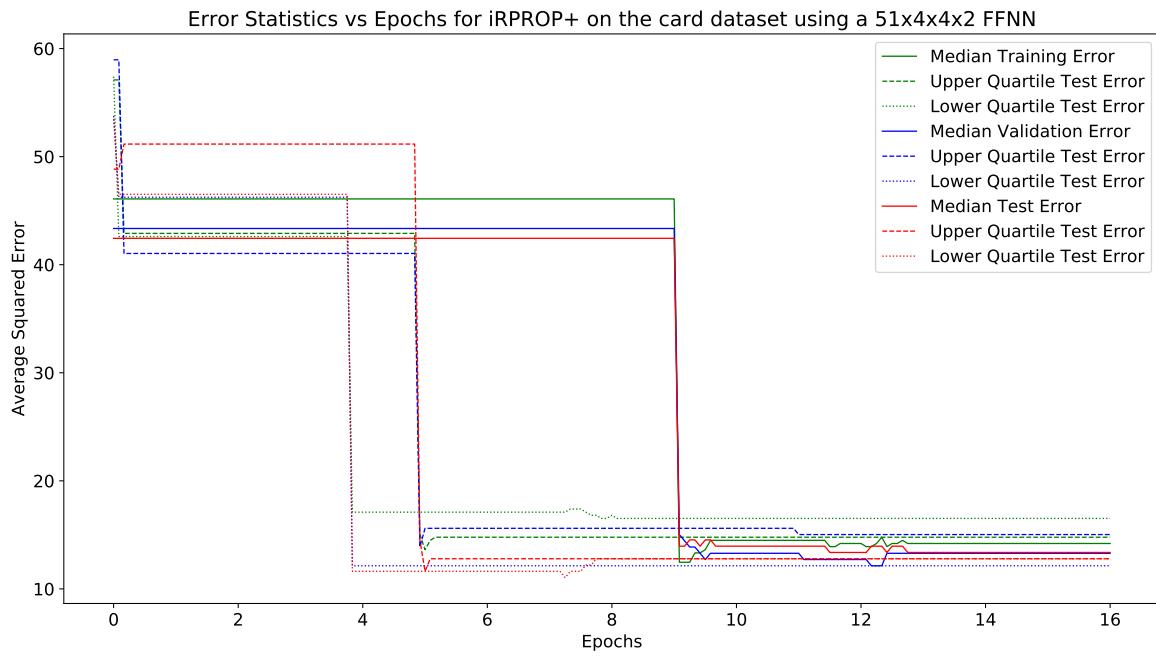
This section displays the results obtained using the card algorithm.

#### 51 × 4 × 4 × 2 Architecture:

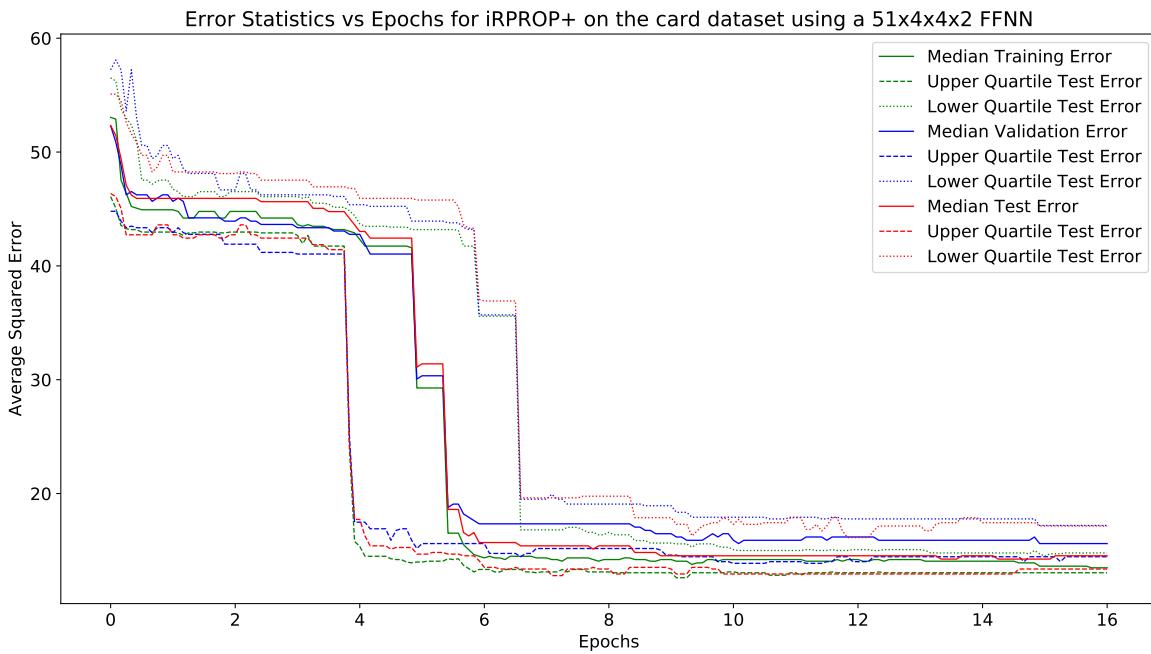
Fig. 191 shows the average and standard deviation of the test, training and validation errors. Fig. 192 shows the median and quartiles of the test, training and validation errors based on the best epoch. Finally, Fig. 193 shows the median and quartiles of the test, training and validation errors per mini-batch.



**Figure 191.** Graph of mean and standard deviation of errors vs epochs



**Figure 192.** Graph of test error vs epochs for the gradient based algorithms



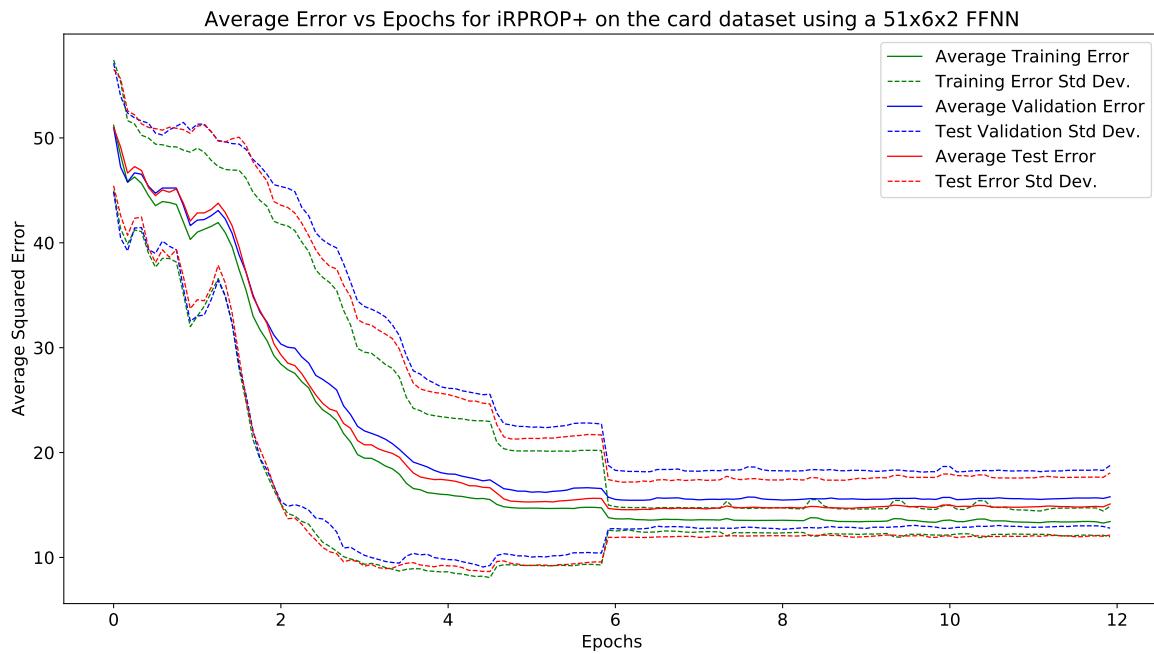
**Figure 193.** Graph of test error vs epochs for the gradient based algorithms

### 16.2.24 card

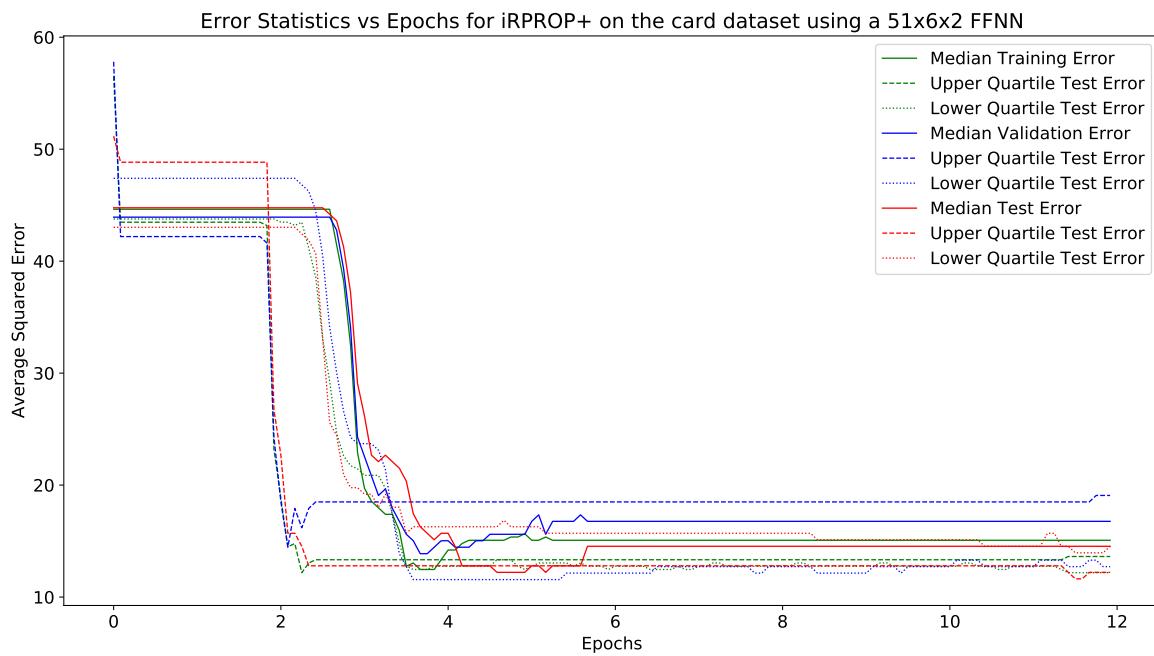
This section displays the results obtained using the card algorithm.

#### 51 × 6 × 2 Architecture:

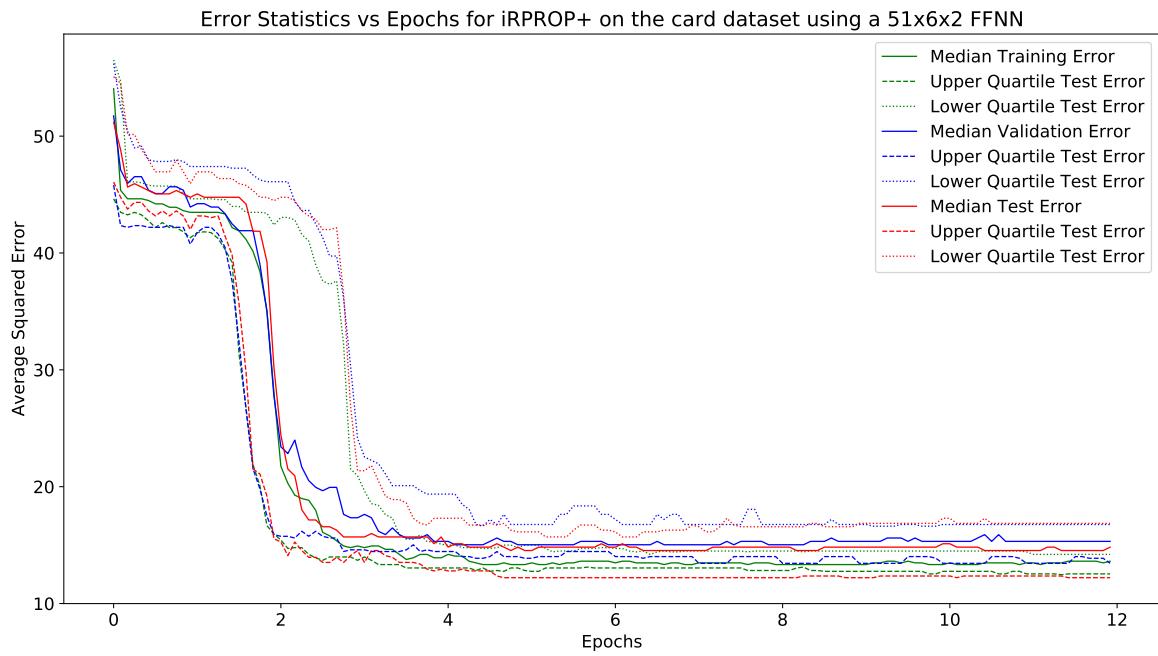
Fig. 194 shows the average and standard deviation of the test, training and validation errors. Fig. 195 shows the median and quartiles of the test, training and validation errors based on the best epoch. Finally, Fig. 196 shows the median and quartiles of the test, training and validation errors per mini-batch.



**Figure 194.** Graph of mean and standard deviation of errors vs epochs



**Figure 195.** Graph of test error vs epochs for the gradient based algorithms



**Figure 196.** Graph of test error vs epochs for the gradient based algorithms

### 16.2.25 PSO

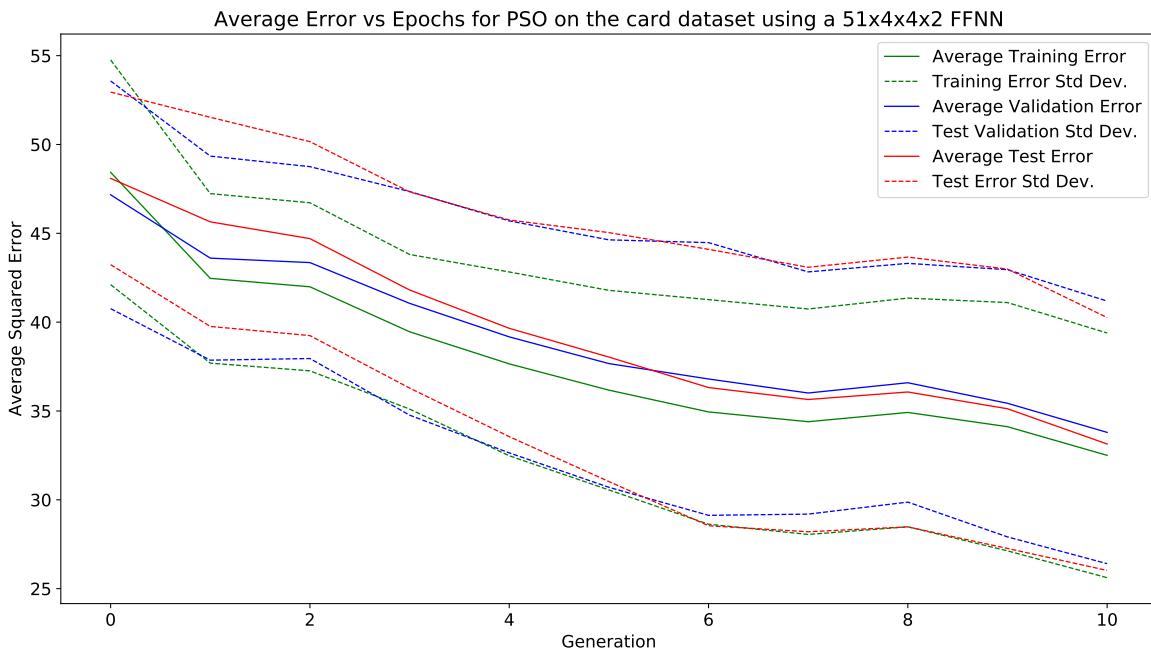
This section displays the results obtained using the PSO algorithm.

### 16.2.26 card

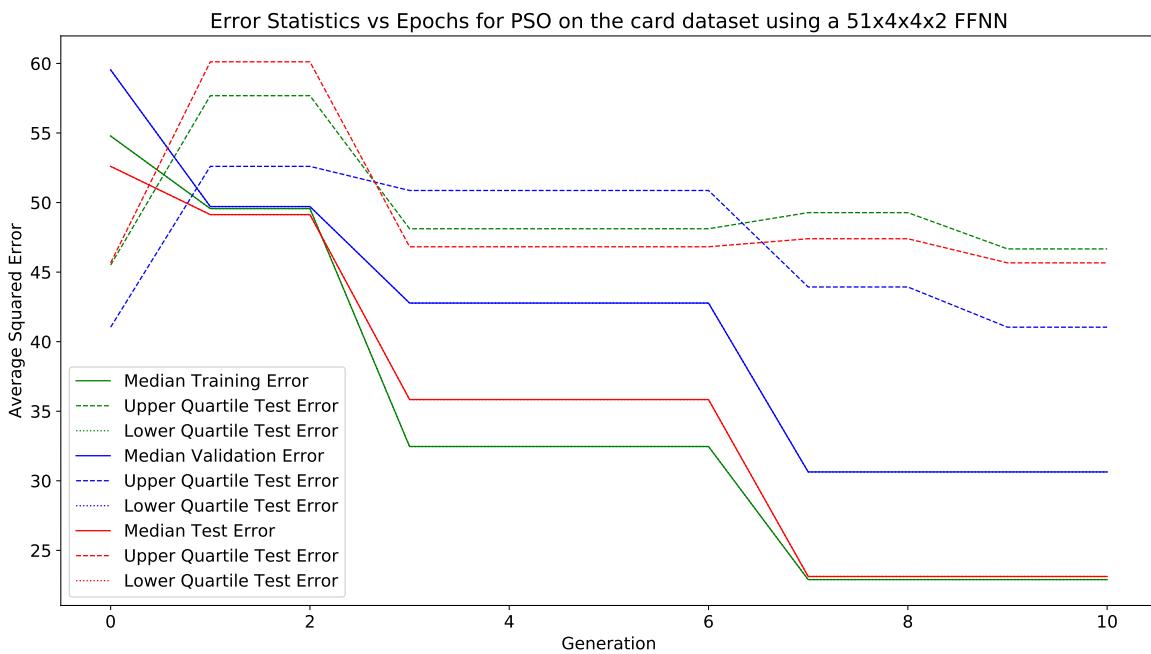
This section displays the results obtained using the card algorithm.

#### 51 × 4 × 4 × 2 Architecture:

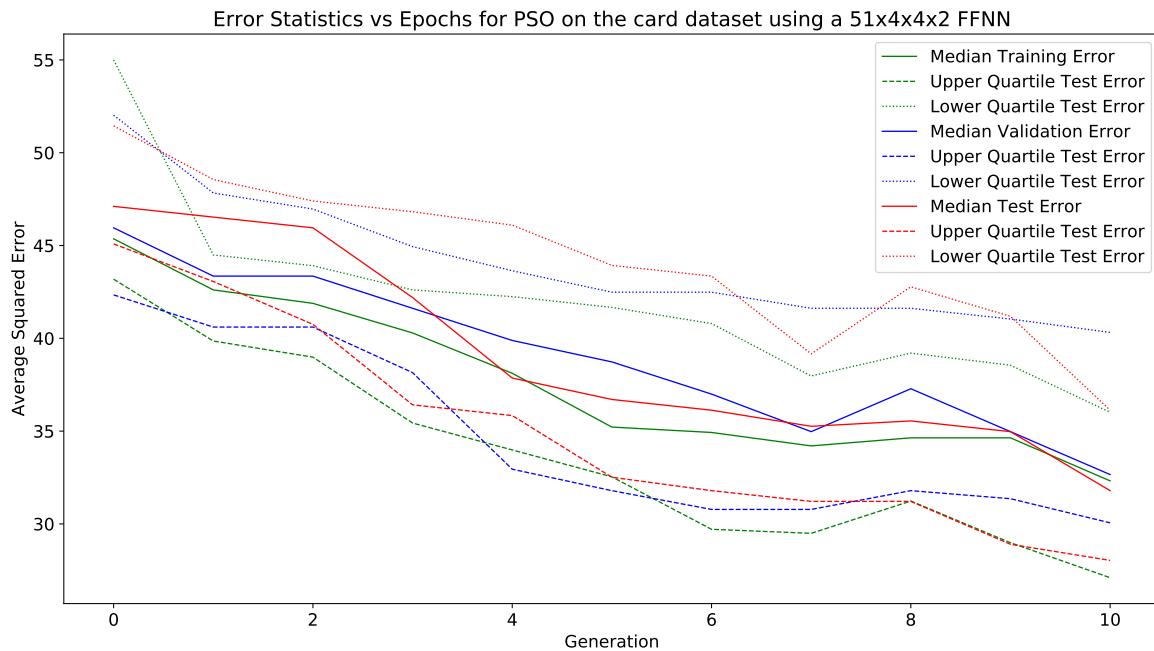
Fig. 197 shows the average and standard deviation of the test, training and validation errors. Fig. 198 shows the median and quartiles of the test, training and validation errors based on the best epoch. Finally, Fig. 199 shows the median and quartiles of the test, training and validation errors per mini-batch.



**Figure 197.** Graph of mean and standard deviation of errors vs epochs



**Figure 198.** Graph of test error vs epochs for the gradient based algorithms



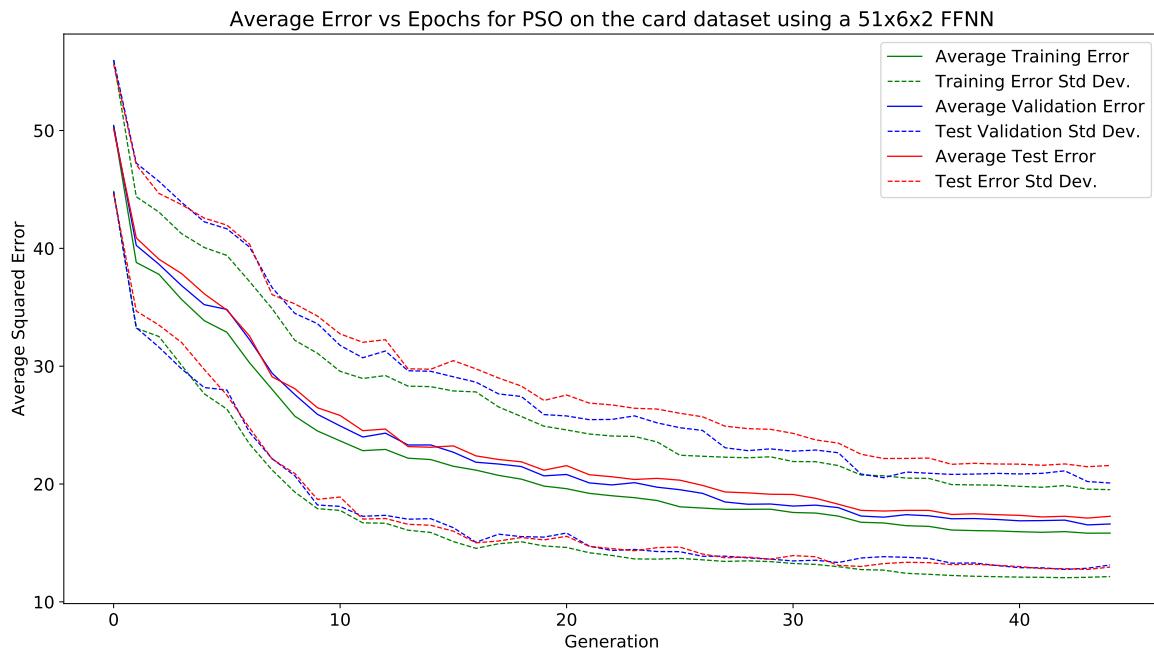
**Figure 199.** Graph of test error vs epochs for the gradient based algorithms

### 16.2.27 card

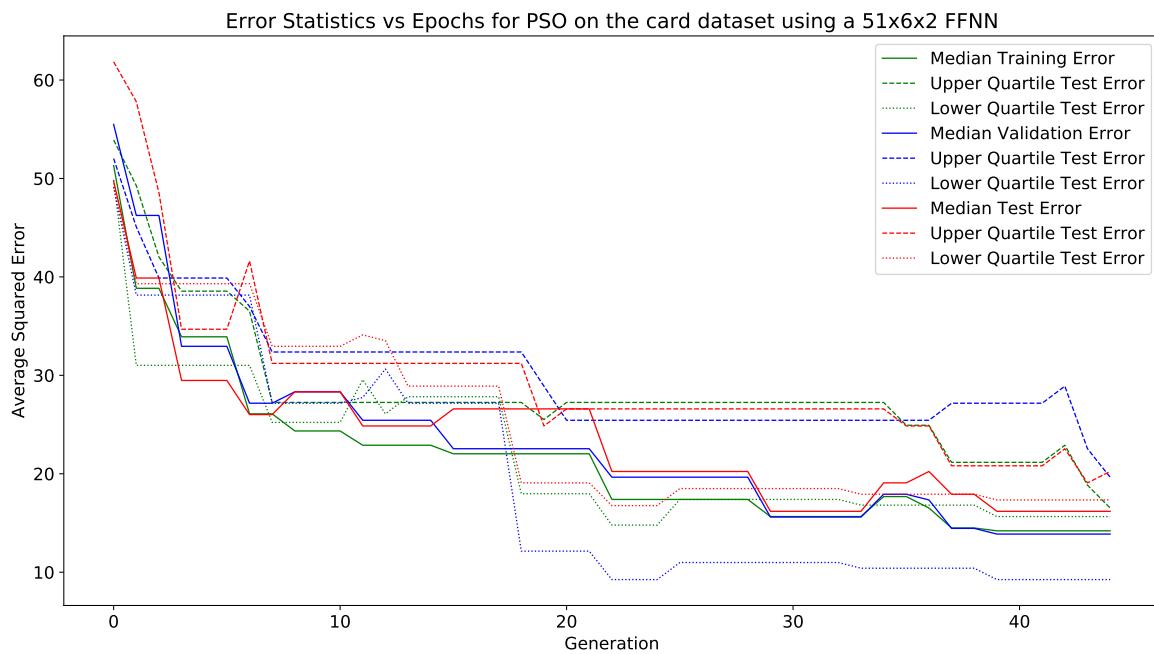
This section displays the results obtained using the card algorithm.

#### 51 × 6 × 2 Architecture:

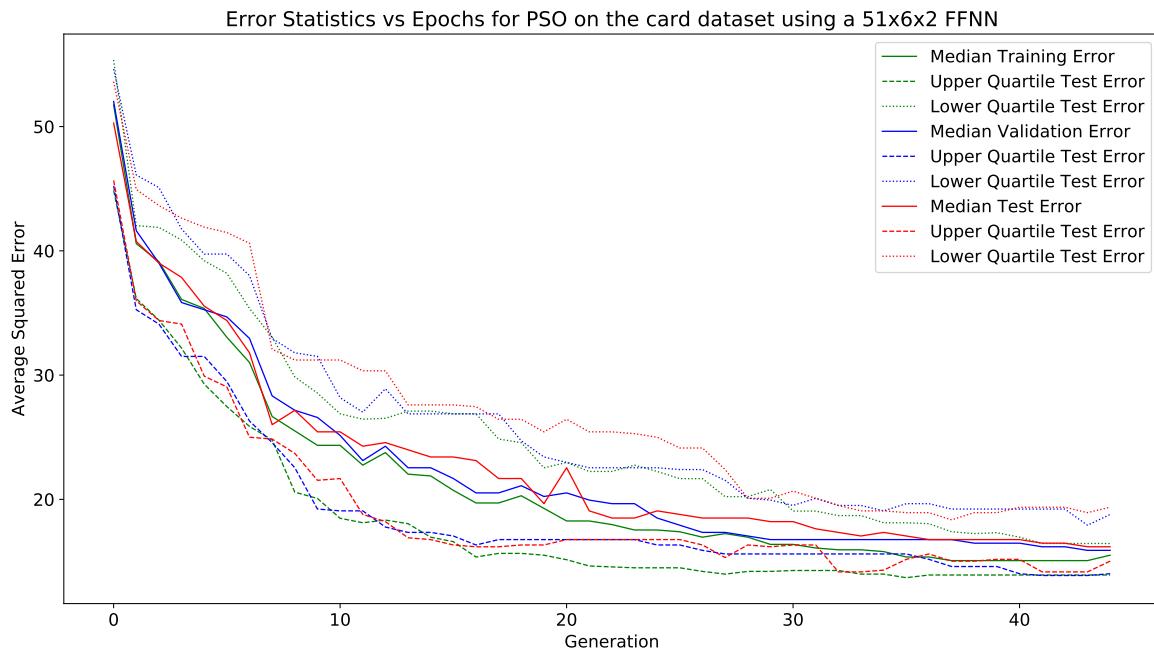
Fig. 200 shows the average and standard deviation of the test, training and validation errors. Fig. 201 shows the median and quartiles of the test, training and validation errors based on the best epoch. Finally, Fig. 202 shows the median and quartiles of the test, training and validation errors per mini-batch.



**Figure 200.** Graph of mean and standard deviation of errors vs epochs



**Figure 201.** Graph of test error vs epochs for the gradient based algorithms



**Figure 202.** Graph of test error vs epochs for the gradient based algorithms

### 16.2.28 QuickProp

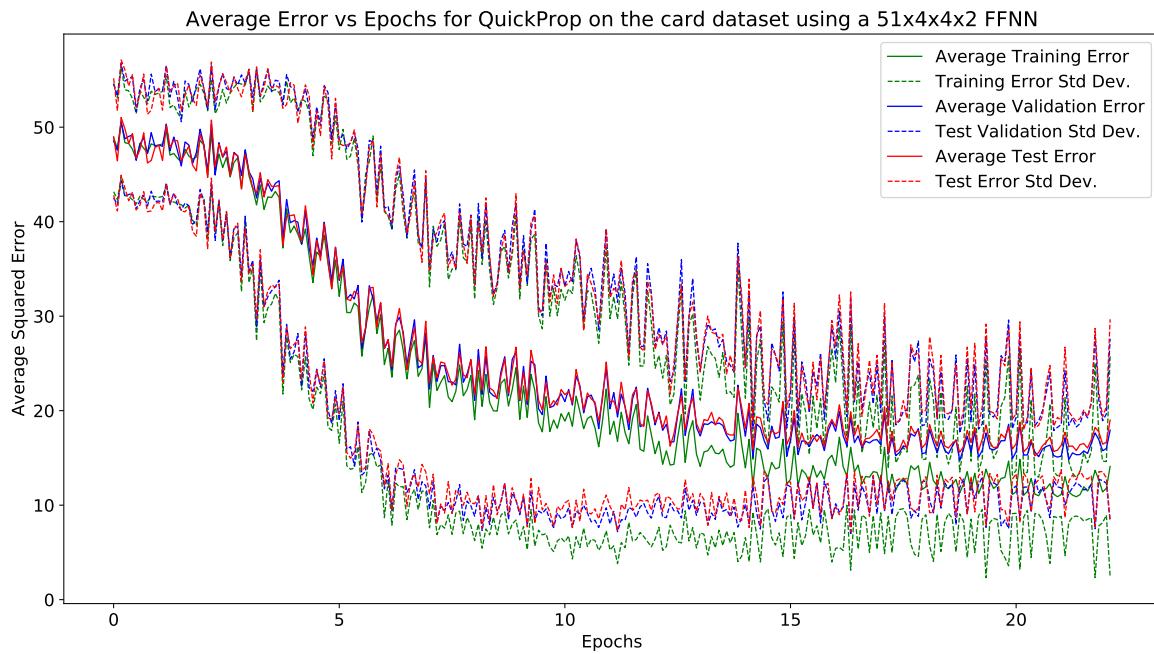
This section displays the results obtained using the QuickProp algorithm.

### 16.2.29 card

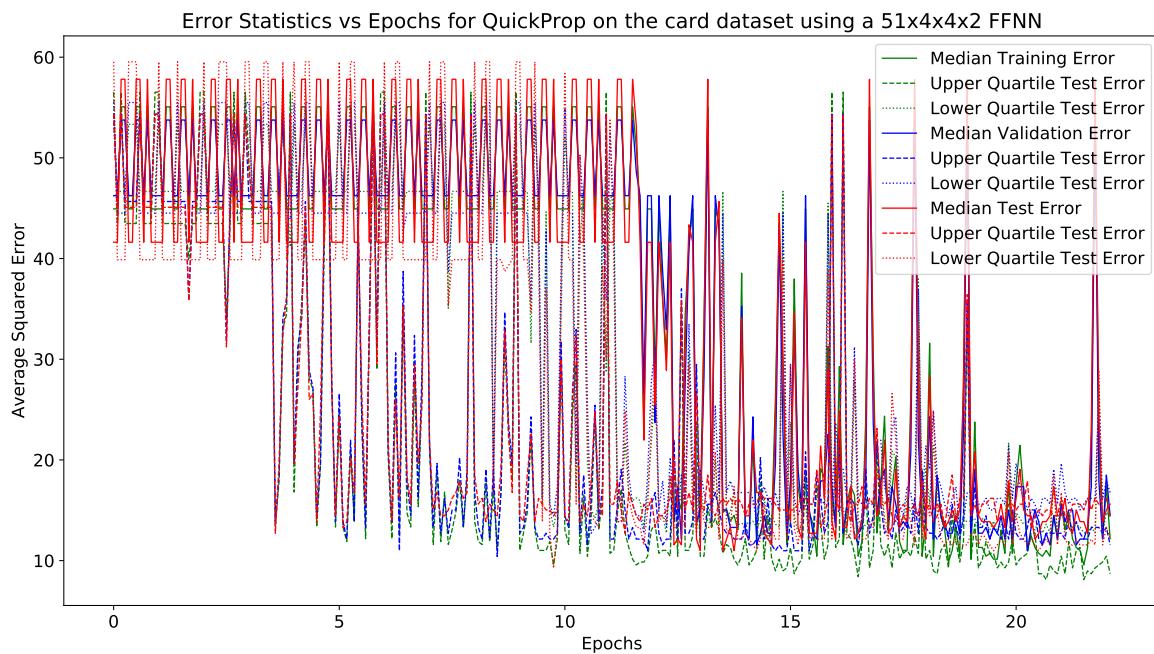
This section displays the results obtained using the card algorithm.

#### 51 × 4 × 4 × 2 Architecture:

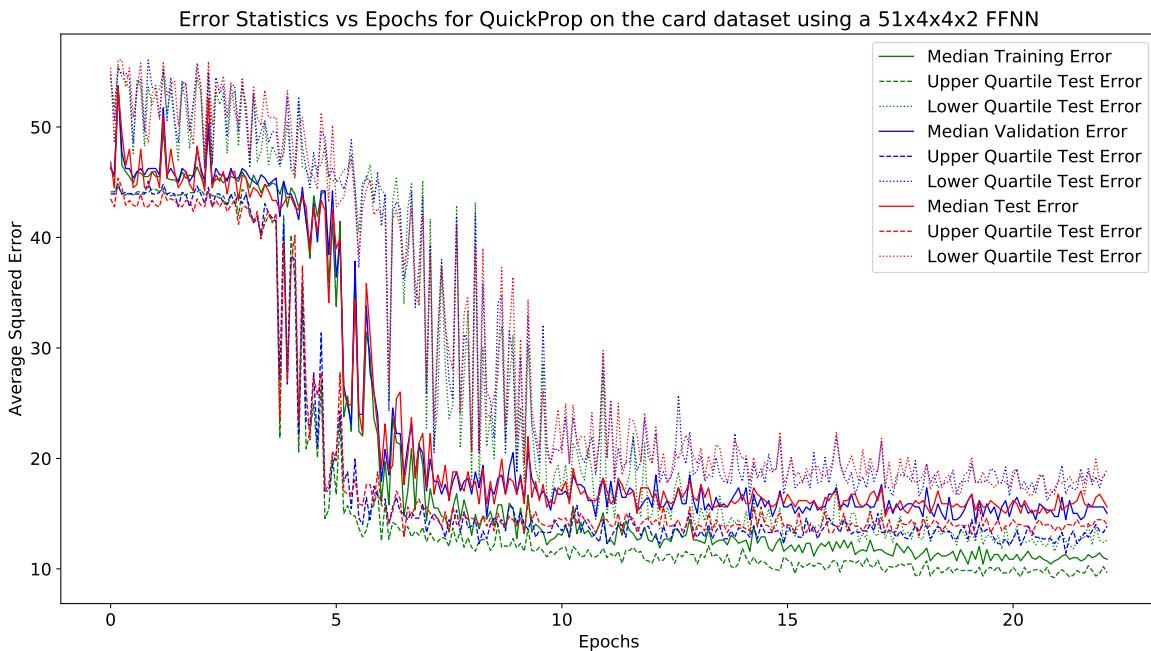
Fig. 203 shows the average and standard deviation of the test, training and validation errors. Fig. 204 shows the median and quartiles of the test, training and validation errors based on the best epoch. Finally, Fig. 205 shows the median and quartiles of the test, training and validation errors per mini-batch.



**Figure 203.** Graph of mean and standard deviation of errors vs epochs



**Figure 204.** Graph of test error vs epochs for the gradient based algorithms



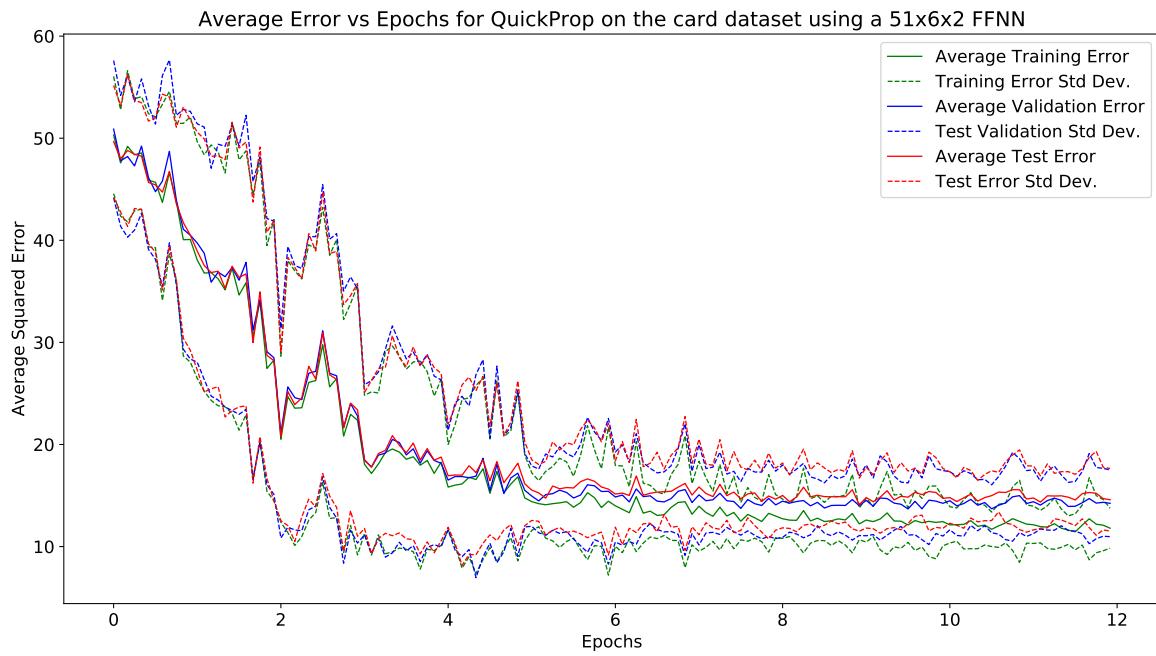
**Figure 205.** Graph of test error vs epochs for the gradient based algorithms

### 16.2.30 card

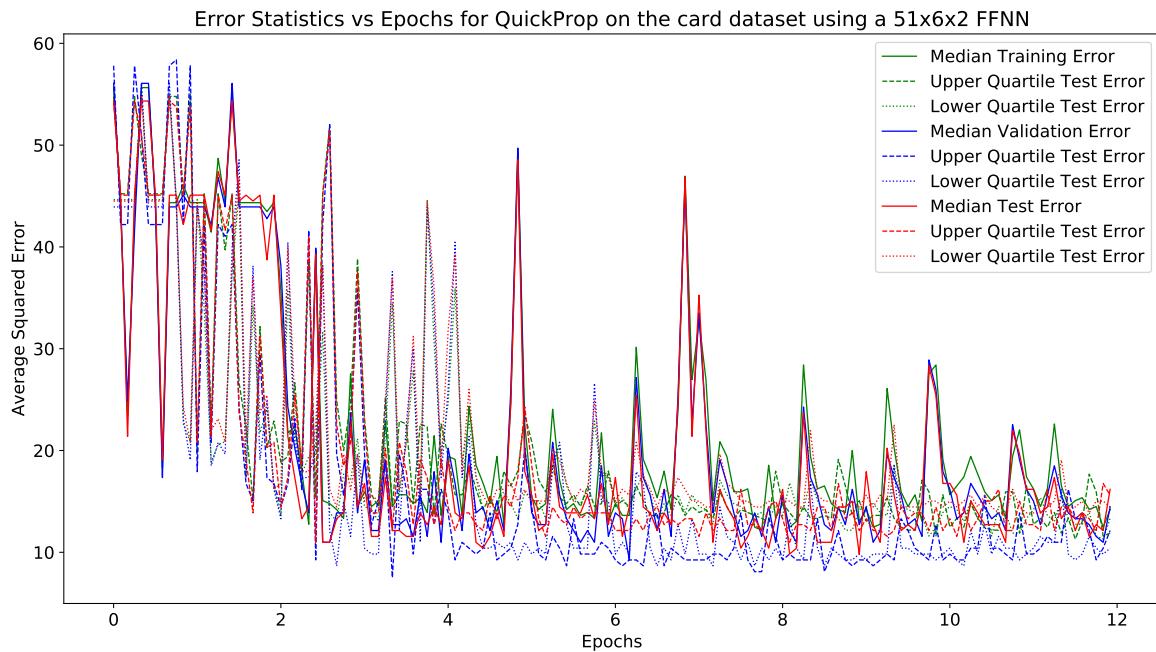
This section displays the results obtained using the card algorithm.

#### 51 × 6 × 2 Architecture:

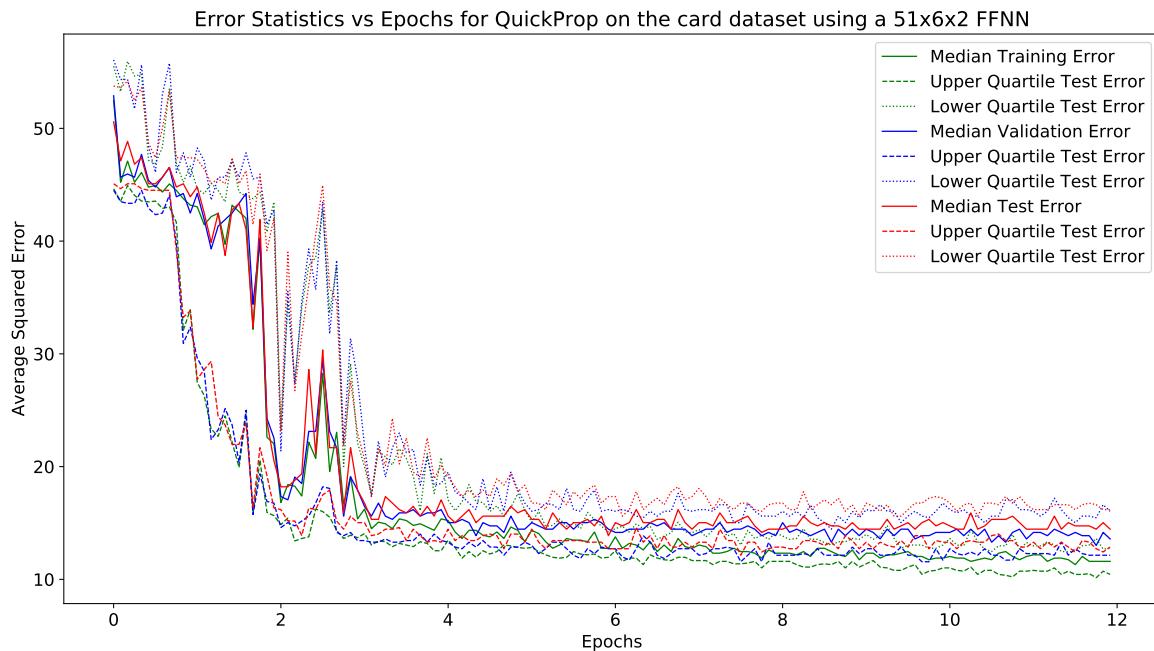
Fig. 206 shows the average and standard deviation of the test, training and validation errors. Fig. 207 shows the median and quartiles of the test, training and validation errors based on the best epoch. Finally, Fig. 208 shows the median and quartiles of the test, training and validation errors per mini-batch.



**Figure 206.** Graph of mean and standard deviation of errors vs epochs



**Figure 207.** Graph of test error vs epochs for the gradient based algorithms



**Figure 208.** Graph of test error vs epochs for the gradient based algorithms

### 16.2.31 RPROP-

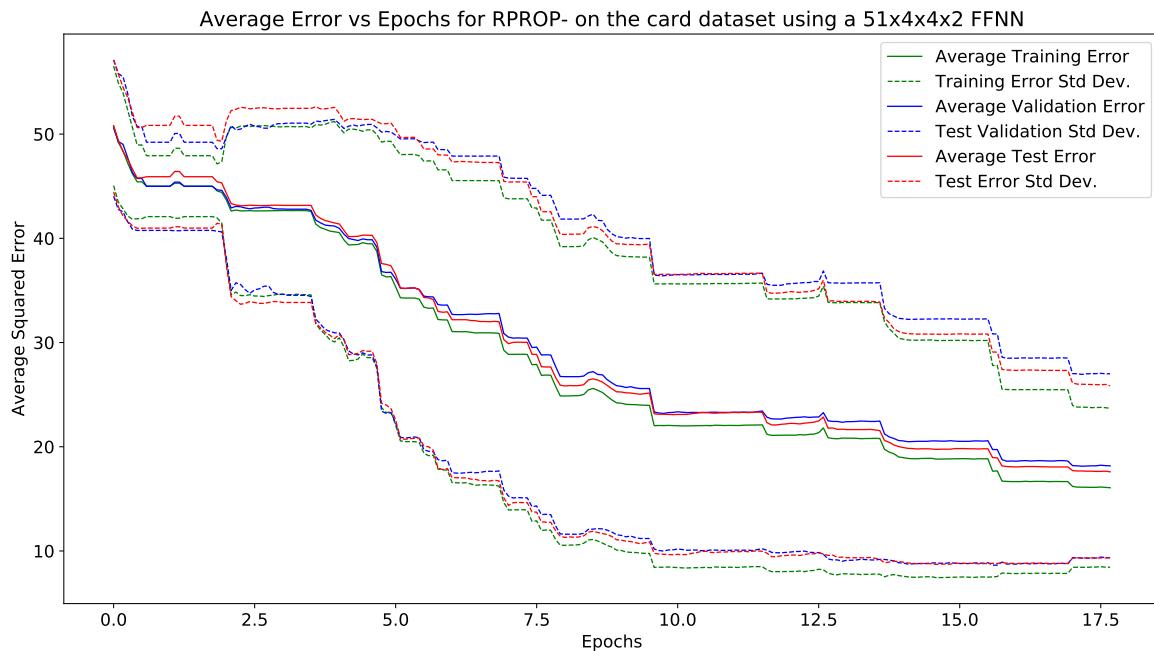
This section displays the results obtained using the RPROP- algorithm.

### 16.2.32 card

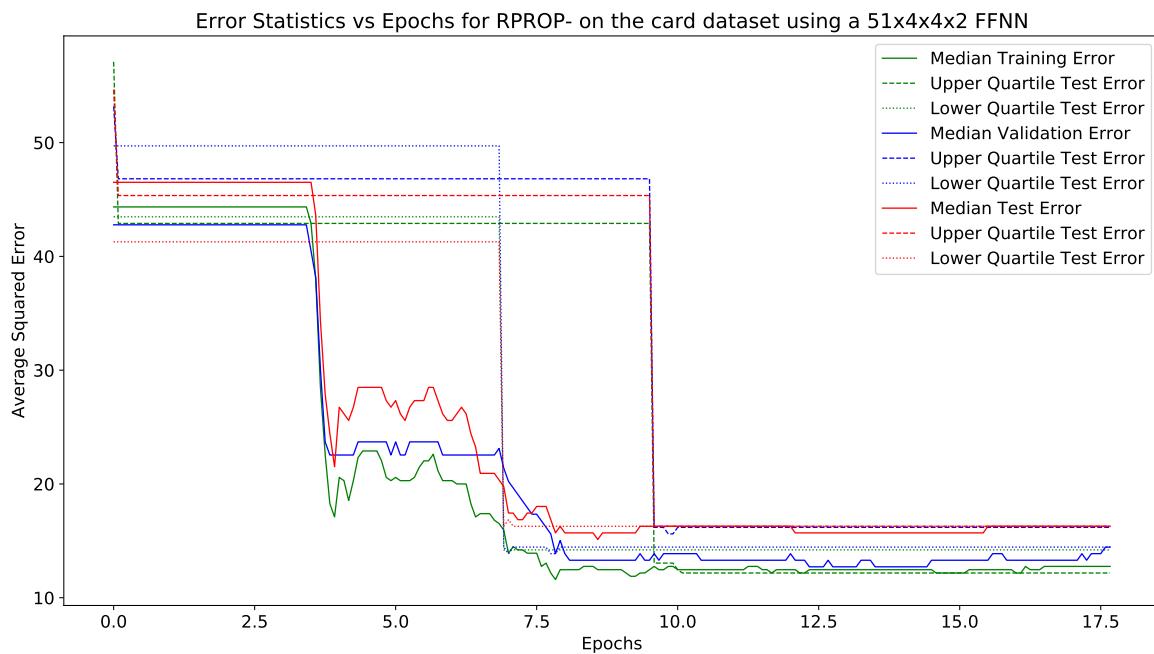
This section displays the results obtained using the card algorithm.

#### 51 × 4 × 4 × 2 Architecture:

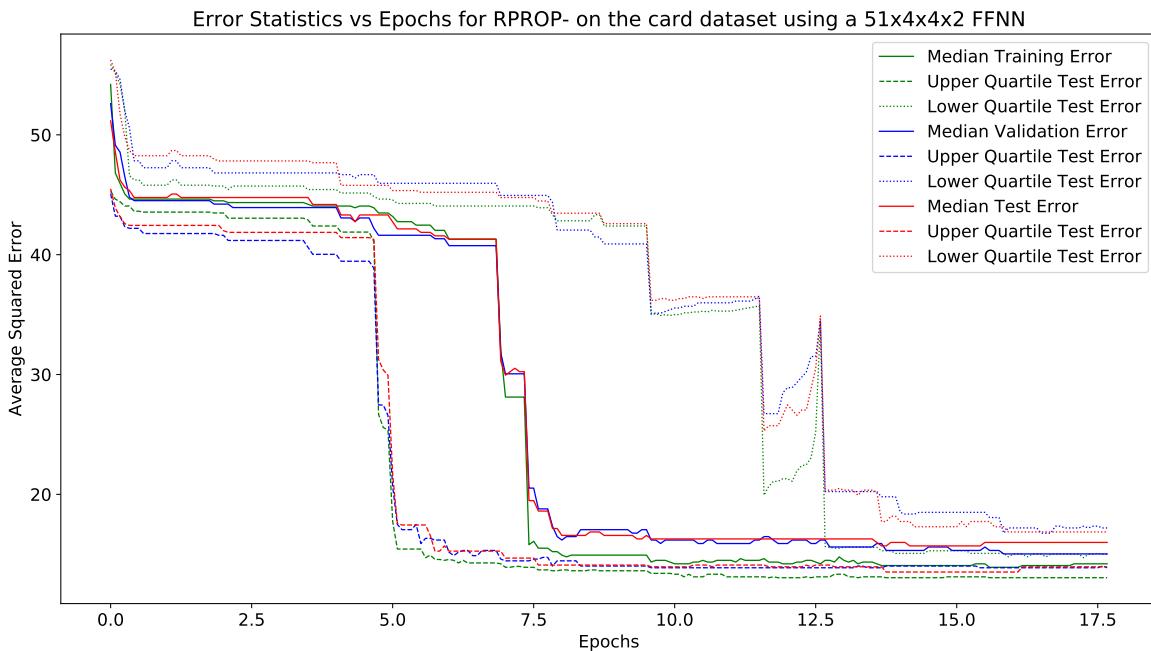
Fig. 209 shows the average and standard deviation of the test, training and validation errors. Fig. 210 shows the median and quartiles of the test, training and validation errors based on the best epoch. Finally, Fig. 211 shows the median and quartiles of the test, training and validation errors per mini-batch.



**Figure 209.** Graph of mean and standard deviation of errors vs epochs



**Figure 210.** Graph of test error vs epochs for the gradient based algorithms



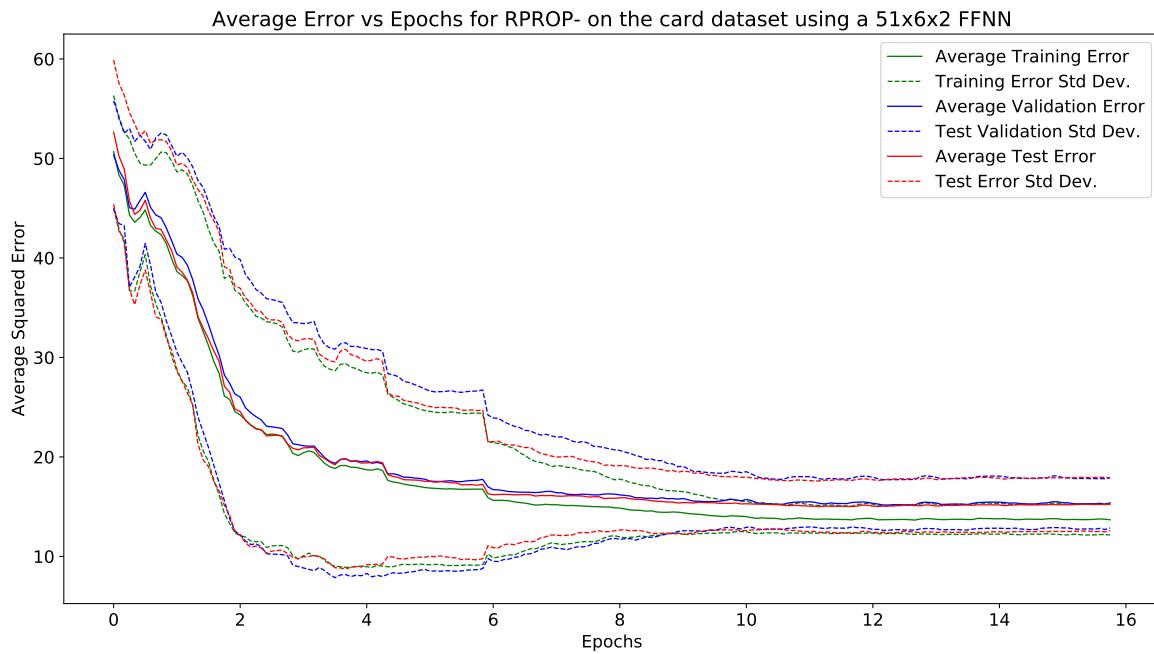
**Figure 211.** Graph of test error vs epochs for the gradient based algorithms

### 16.2.33 card

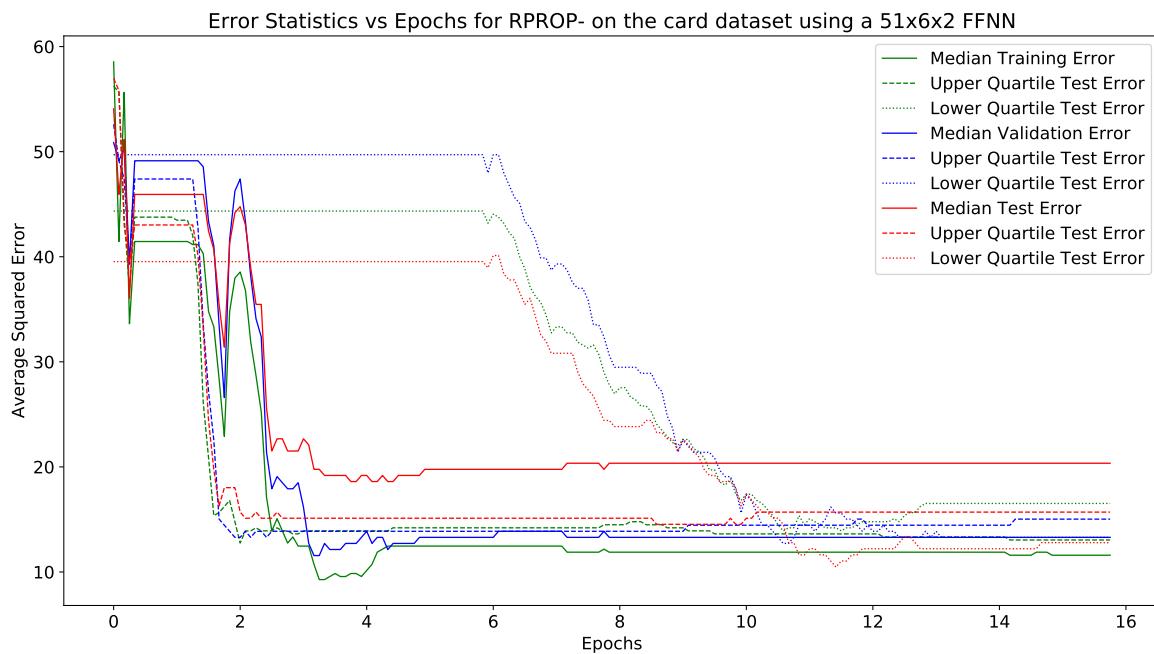
This section displays the results obtained using the card algorithm.

#### 51 × 6 × 2 Architecture:

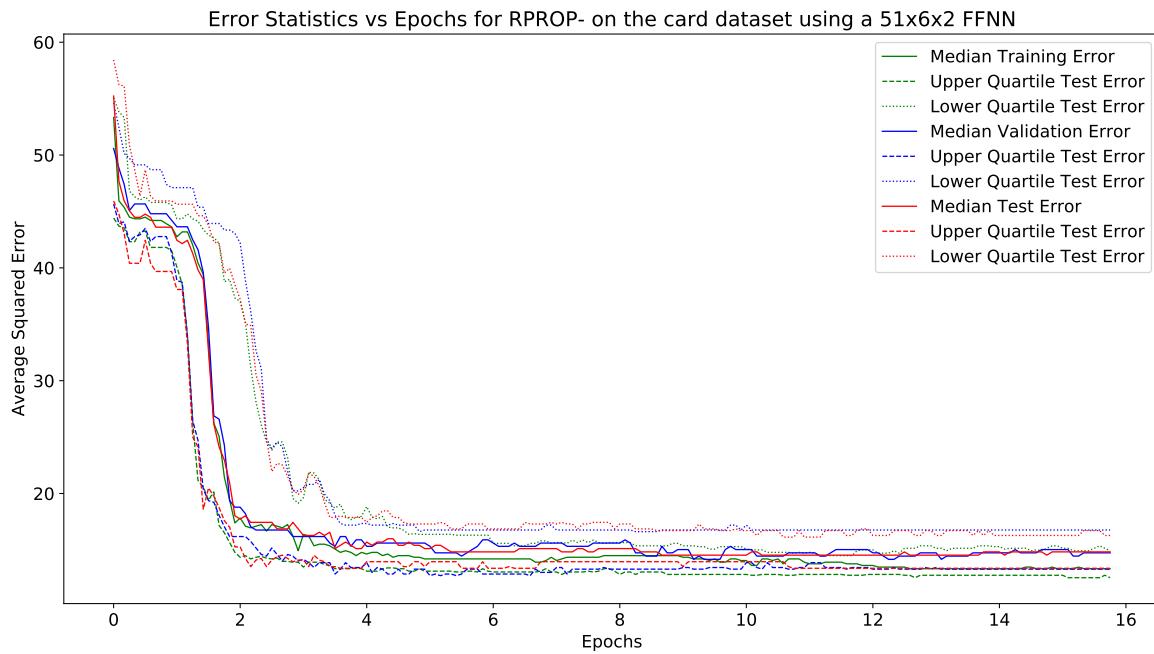
Fig. 212 shows the average and standard deviation of the test, training and validation errors. Fig. 213 shows the median and quartiles of the test, training and validation errors based on the best epoch. Finally, Fig. 214 shows the median and quartiles of the test, training and validation errors per mini-batch.



**Figure 212.** Graph of mean and standard deviation of errors vs epochs



**Figure 213.** Graph of test error vs epochs for the gradient based algorithms



**Figure 214.** Graph of test error vs epochs for the gradient based algorithms

### 16.2.34 RPROP+

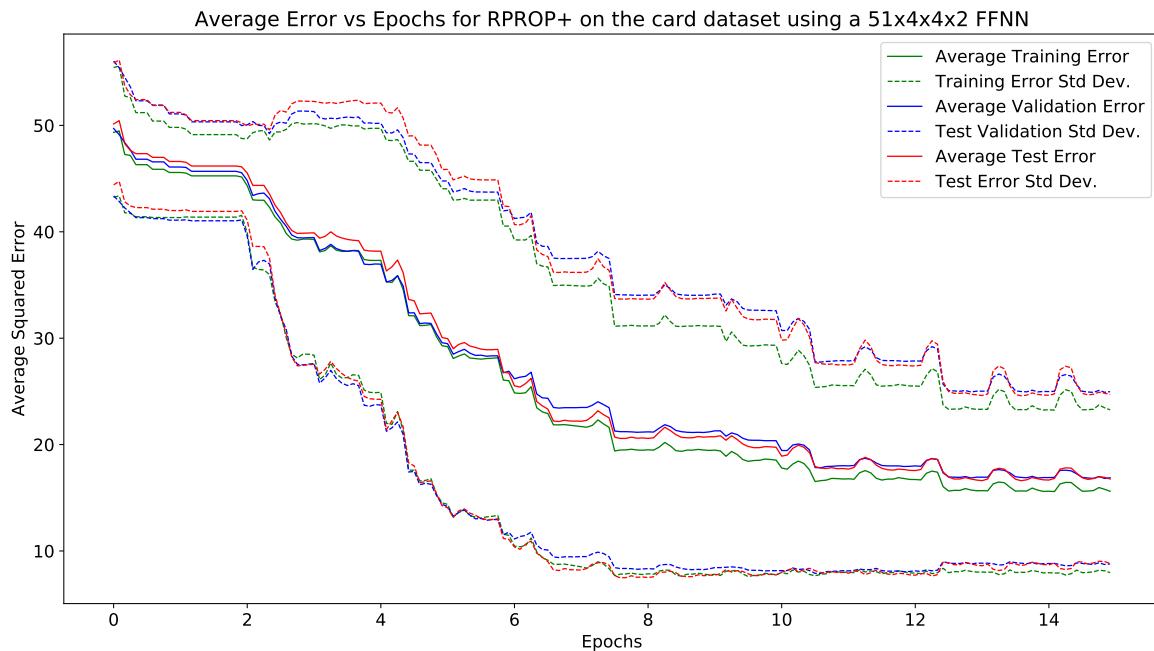
This section displays the results obtained using the RPROP+ algorithm.

### 16.2.35 card

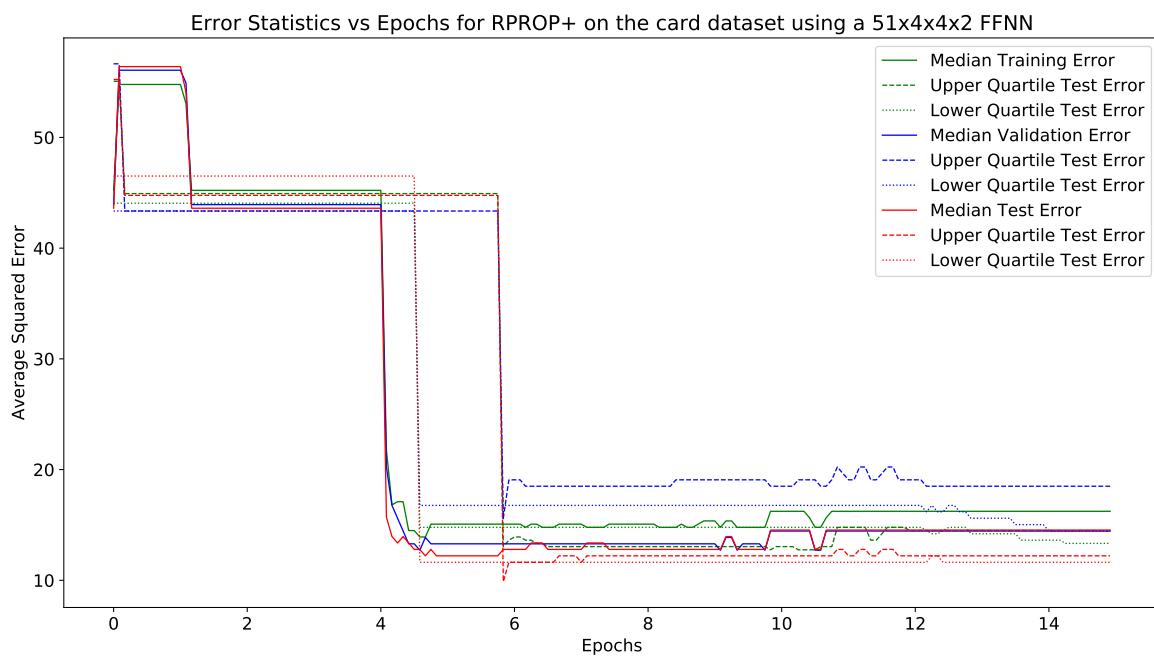
This section displays the results obtained using the card algorithm.

#### 51 × 4 × 4 × 2 Architecture:

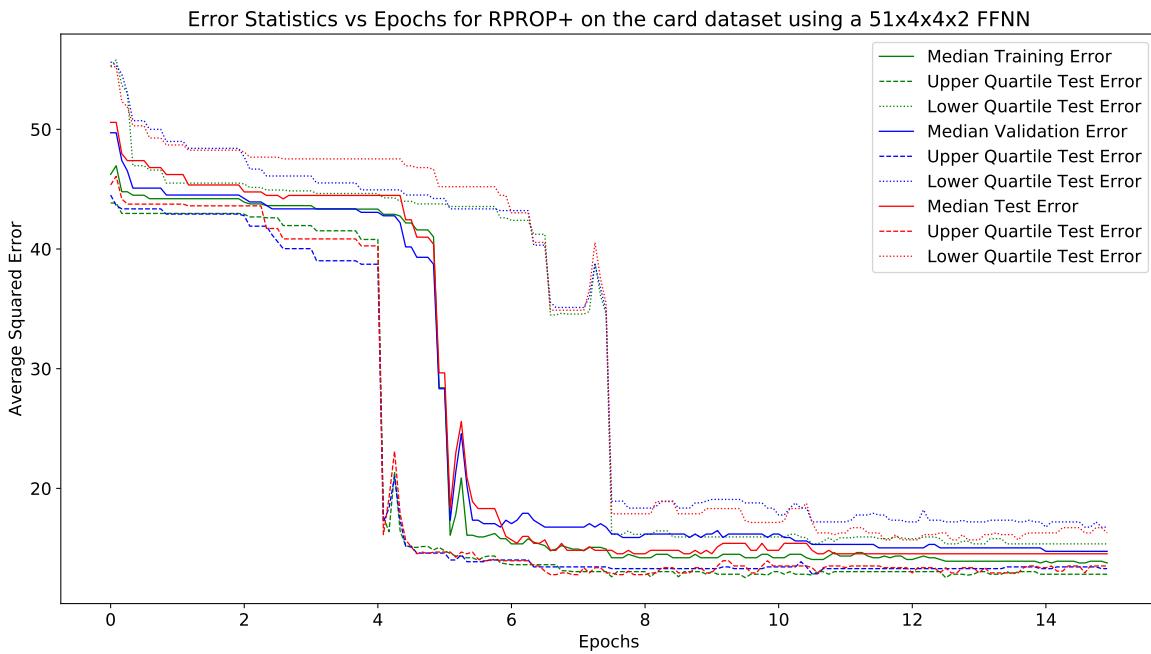
Fig. 215 shows the average and standard deviation of the test, training and validation errors. Fig. 216 shows the median and quartiles of the test, training and validation errors based on the best epoch. Finally, Fig. 217 shows the median and quartiles of the test, training and validation errors per mini-batch.



**Figure 215.** Graph of mean and standard deviation of errors vs epochs



**Figure 216.** Graph of test error vs epochs for the gradient based algorithms



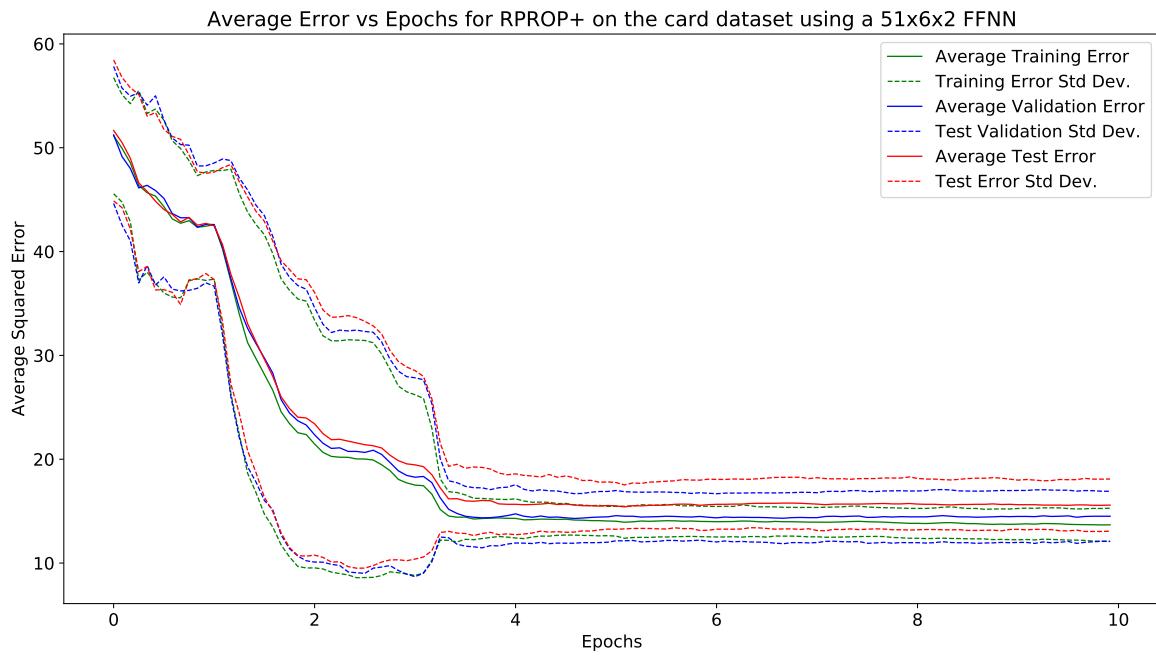
**Figure 217.** Graph of test error vs epochs for the gradient based algorithms

### 16.2.36 card

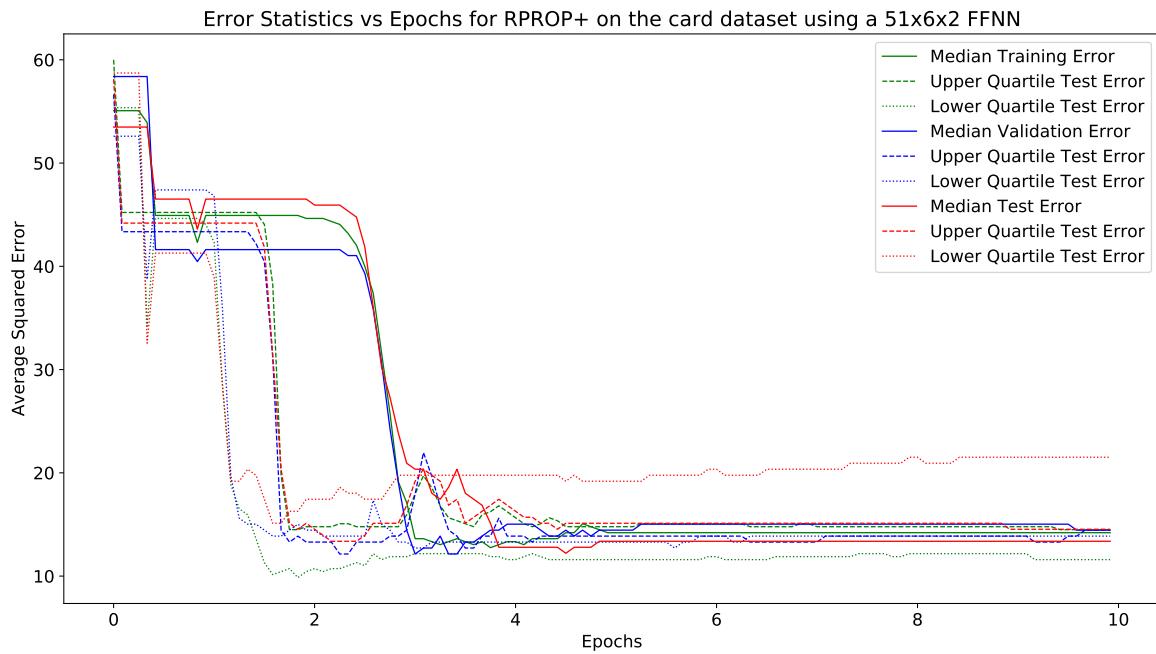
This section displays the results obtained using the card algorithm.

#### 51 × 6 × 2 Architecture:

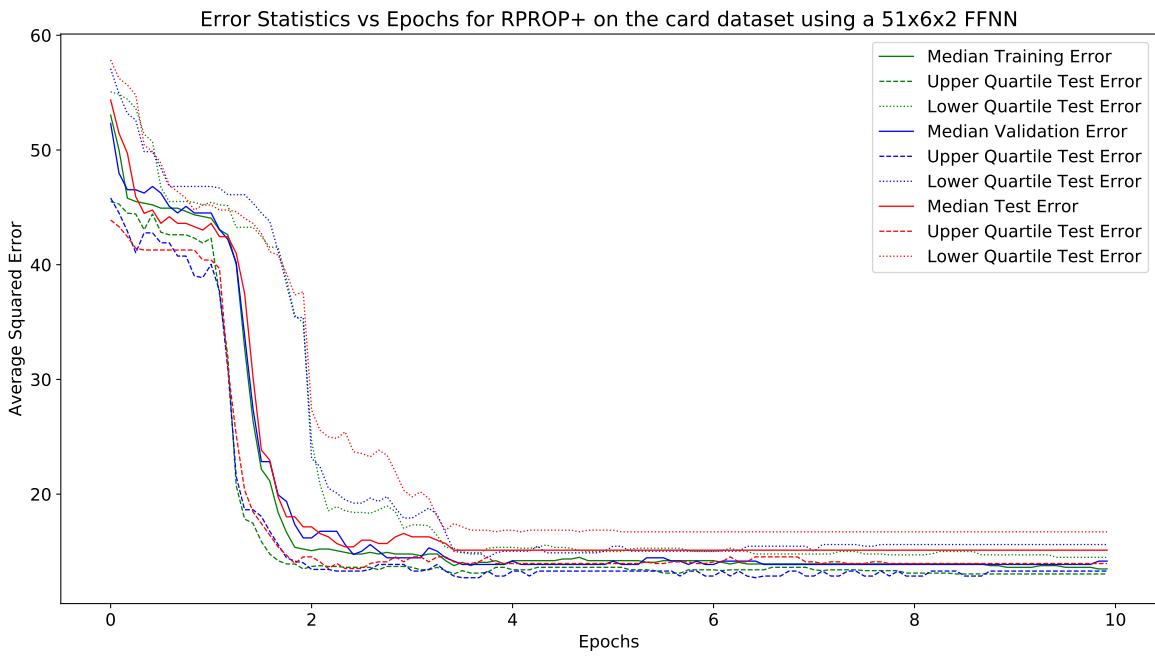
Fig. 218 shows the average and standard deviation of the test, training and validation errors. Fig. 219 shows the median and quartiles of the test, training and validation errors based on the best epoch. Finally, Fig. 220 shows the median and quartiles of the test, training and validation errors per mini-batch.



**Figure 218.** Graph of mean and standard deviation of errors vs epochs



**Figure 219.** Graph of test error vs epochs for the gradient based algorithms



**Figure 220.** Graph of test error vs epochs for the gradient based algorithms

### 16.3 flare

This section displays the results obtained in the flare dataset.

#### 16.3.1 ADAGRAD

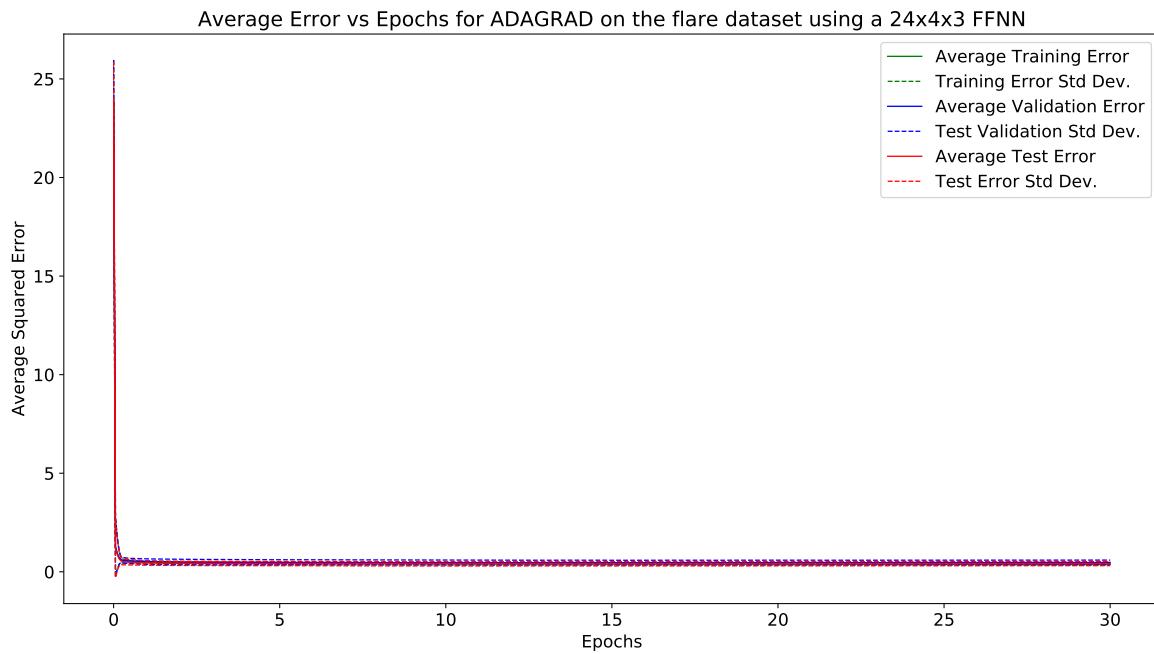
This section displays the results obtained using the ADAGRAD algorithm.

#### 16.3.2 flare

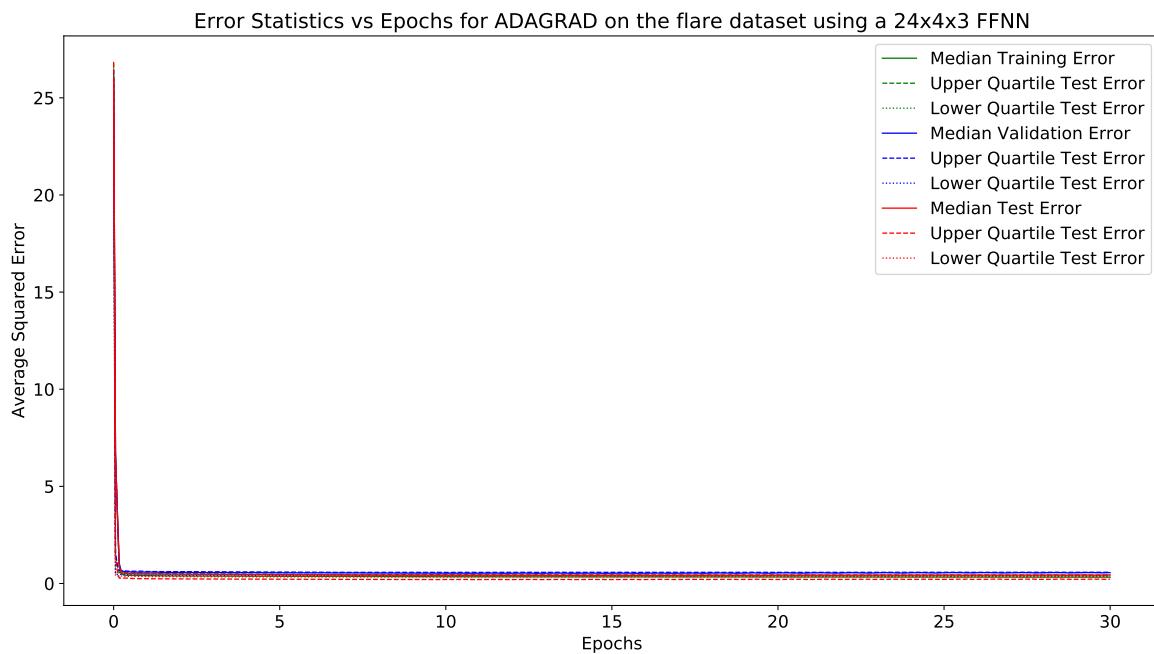
This section displays the results obtained using the flare algorithm.

#### 24 × 4 × 3 Architecture:

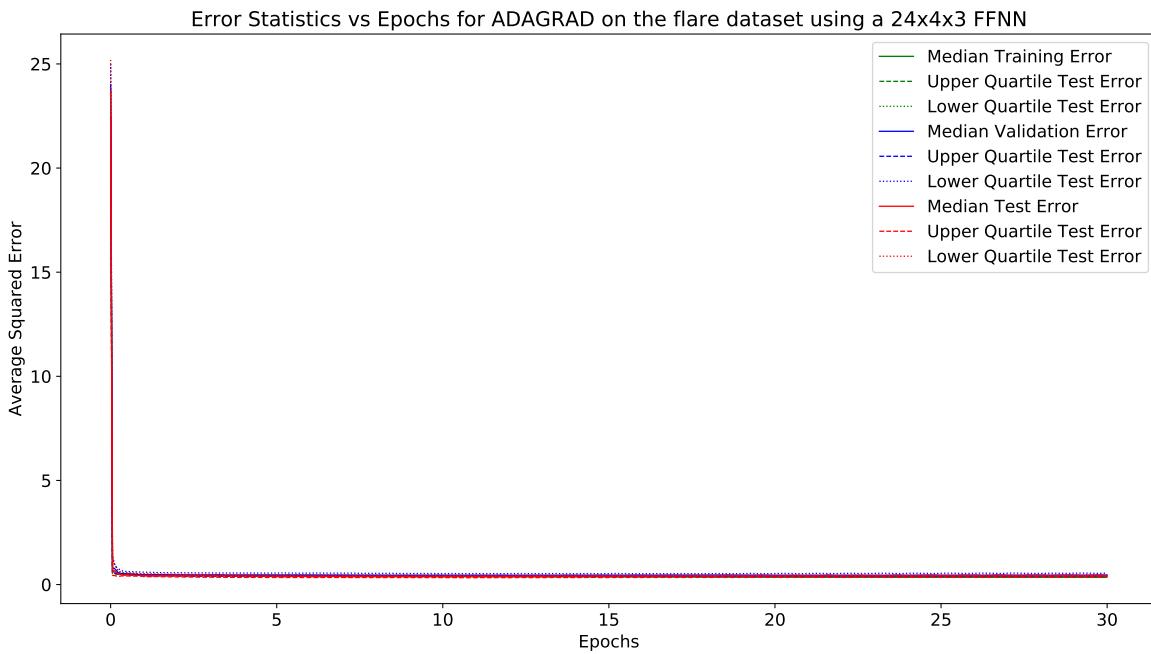
Fig. 221 shows the average and standard deviation of the test, training and validation errors. Fig. 222 shows the median and quartiles of the test, training and validation errors based on the best epoch. Finally, Fig. 223 shows the median and quartiles of the test, training and validation errors per mini-batch.



**Figure 221.** Graph of mean and standard deviation of errors vs epochs



**Figure 222.** Graph of test error vs epochs for the gradient based algorithms



**Figure 223.** Graph of test error vs epochs for the gradient based algorithms

### 16.3.3 Backprop with Momentum

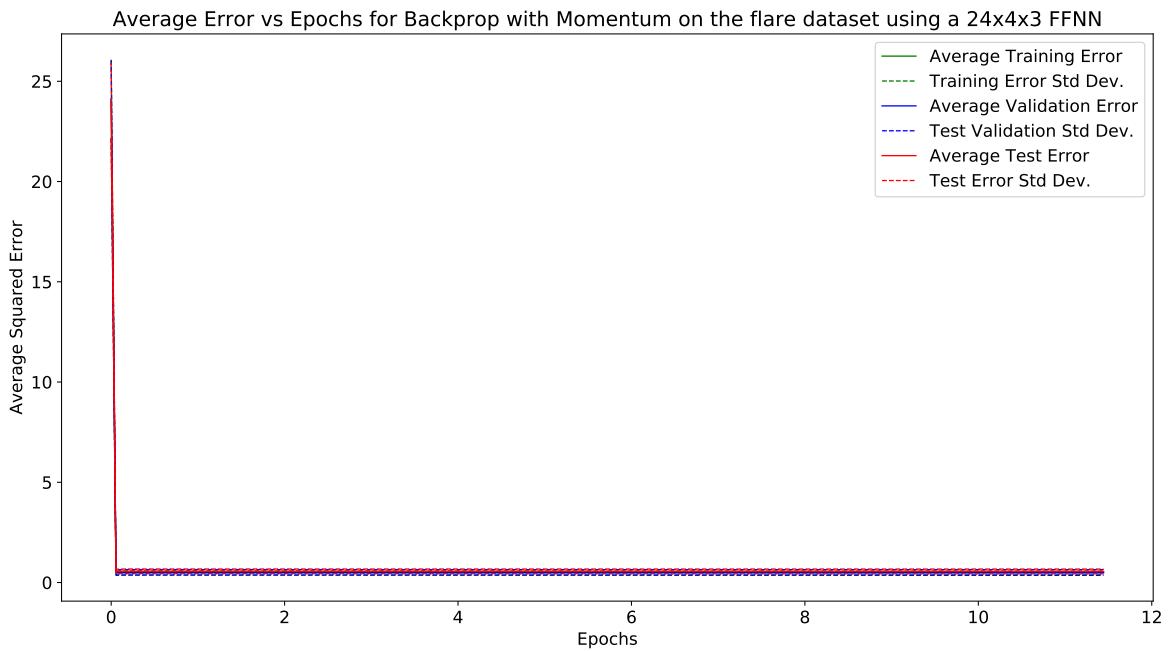
This section displays the results obtained using the Backprop with Momentum algorithm.

### 16.3.4 flare

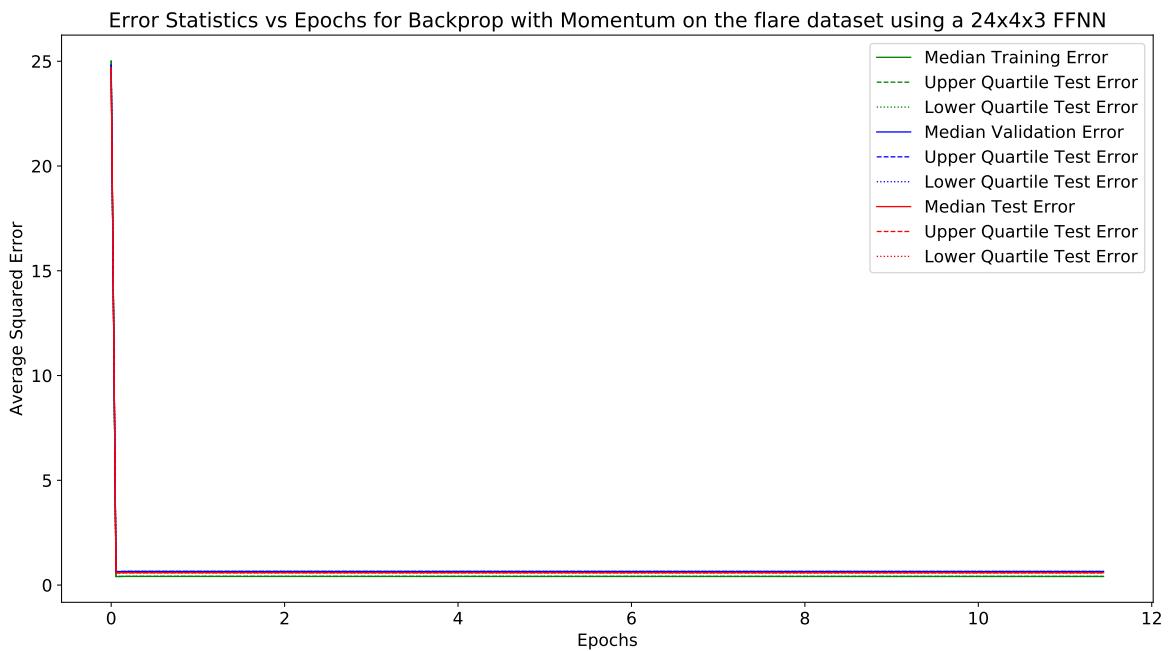
This section displays the results obtained using the flare algorithm.

#### 24 × 4 × 3 Architecture:

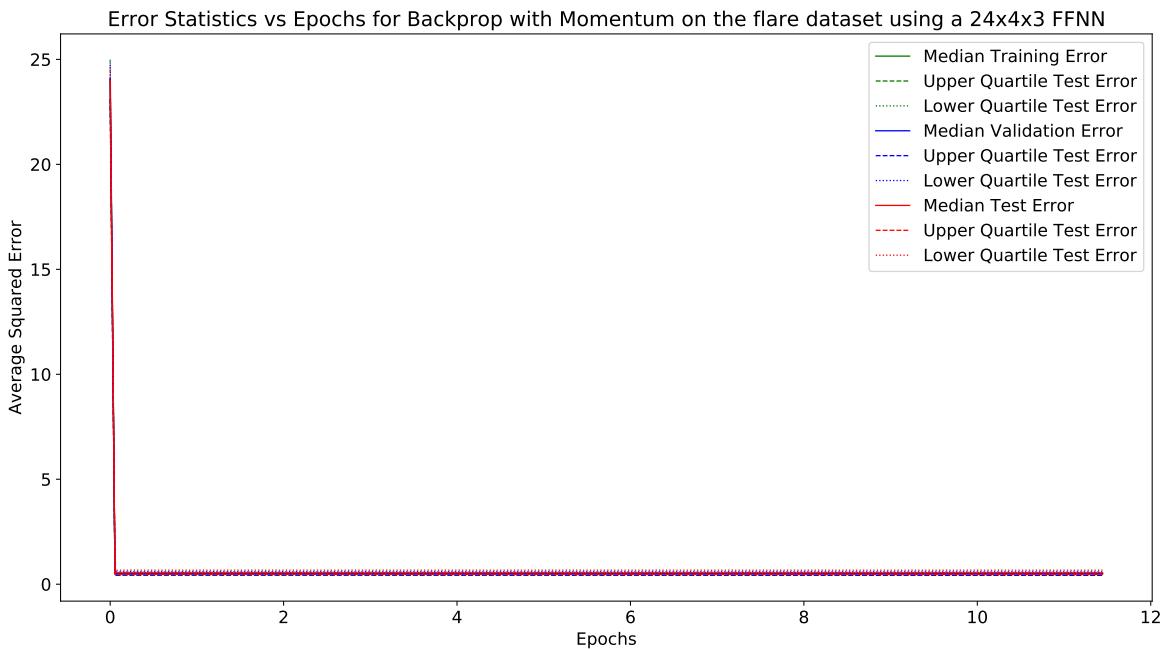
Fig. 224 shows the average and standard deviation of the test, training and validation errors. Fig. 225 shows the median and quartiles of the test, training and validation errors based on the best epoch. Finally, Fig. 226 shows the median and quartiles of the test, training and validation errors per mini-batch.



**Figure 224.** Graph of mean and standard deviation of errors vs epochs



**Figure 225.** Graph of test error vs epochs for the gradient based algorithms



**Figure 226.** Graph of test error vs epochs for the gradient based algorithms

### 16.3.5 back-propogation

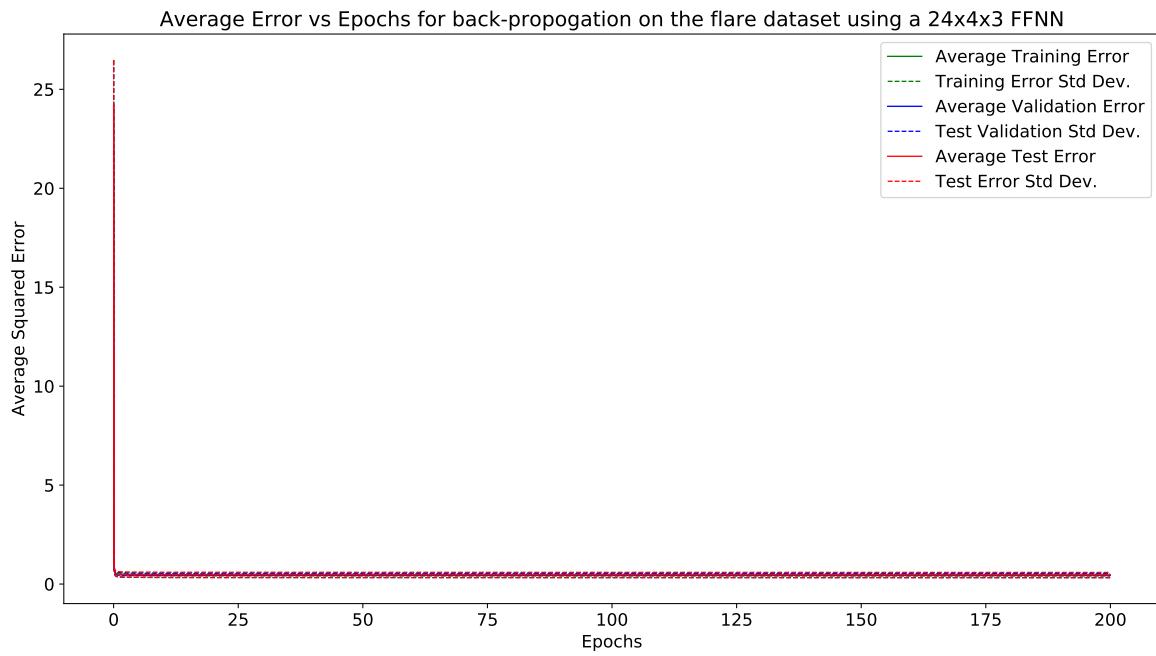
This section displays the results obtained using the back-propogation algorithm.

### 16.3.6 flare

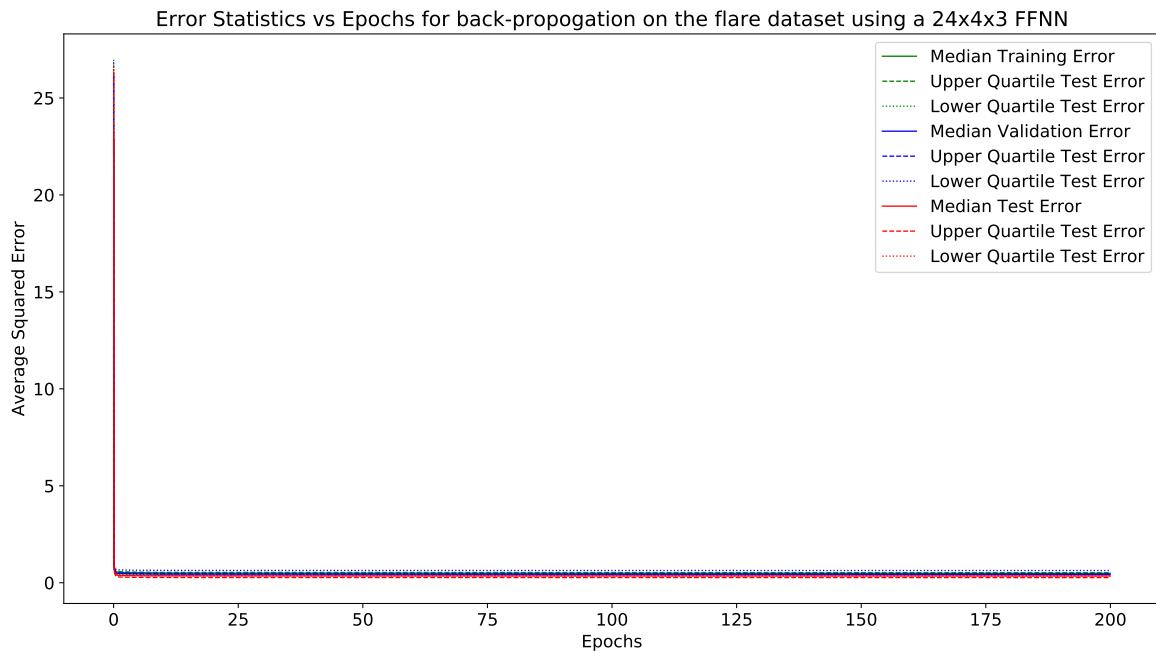
This section displays the results obtained using the flare algorithm.

#### 24 × 4 × 3 Architecture:

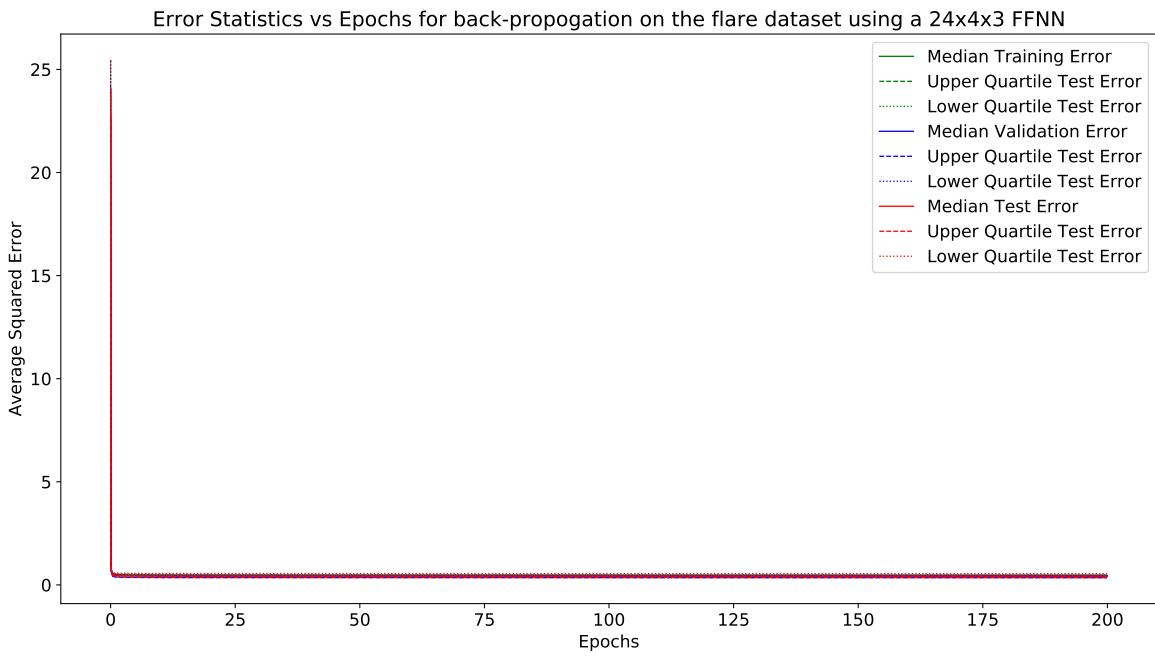
Fig. 227 shows the average and standard deviation of the test, training and validation errors. Fig. 228 shows the median and quartiles of the test, training and validation errors based on the best epoch. Finally, Fig. 229 shows the median and quartiles of the test, training and validation errors per mini-batch.



**Figure 227.** Graph of mean and standard deviation of errors vs epochs



**Figure 228.** Graph of test error vs epochs for the gradient based algorithms



**Figure 229.** Graph of test error vs epochs for the gradient based algorithms

### 16.3.7 GA1

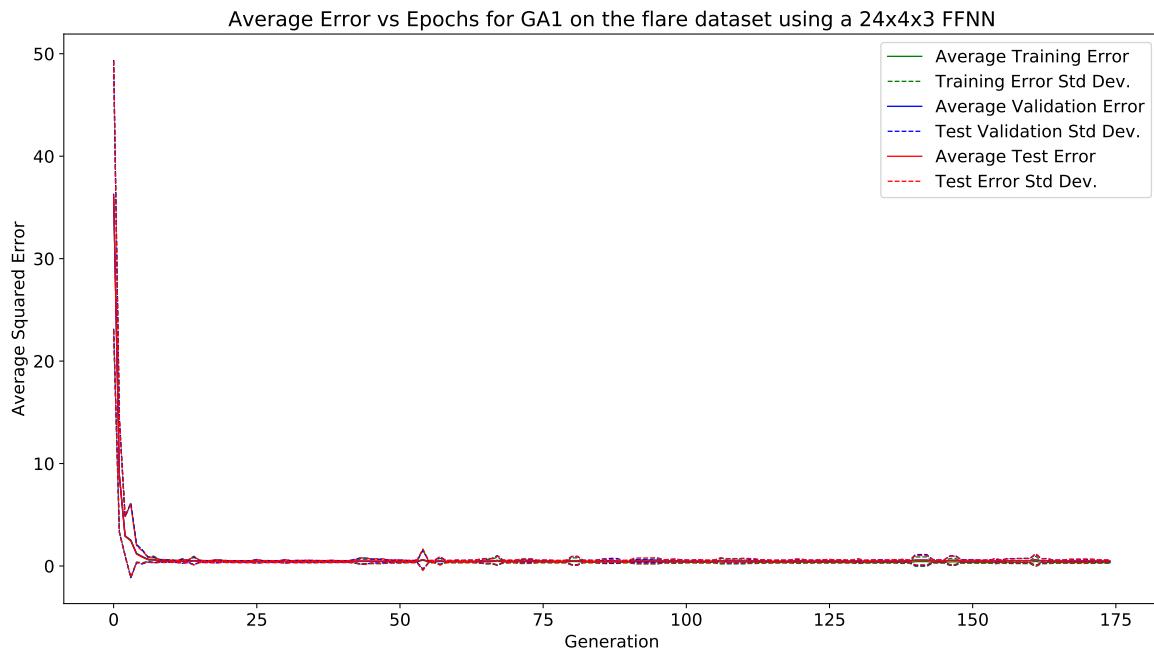
This section displays the results obtained using the GA1 algorithm.

### 16.3.8 flare

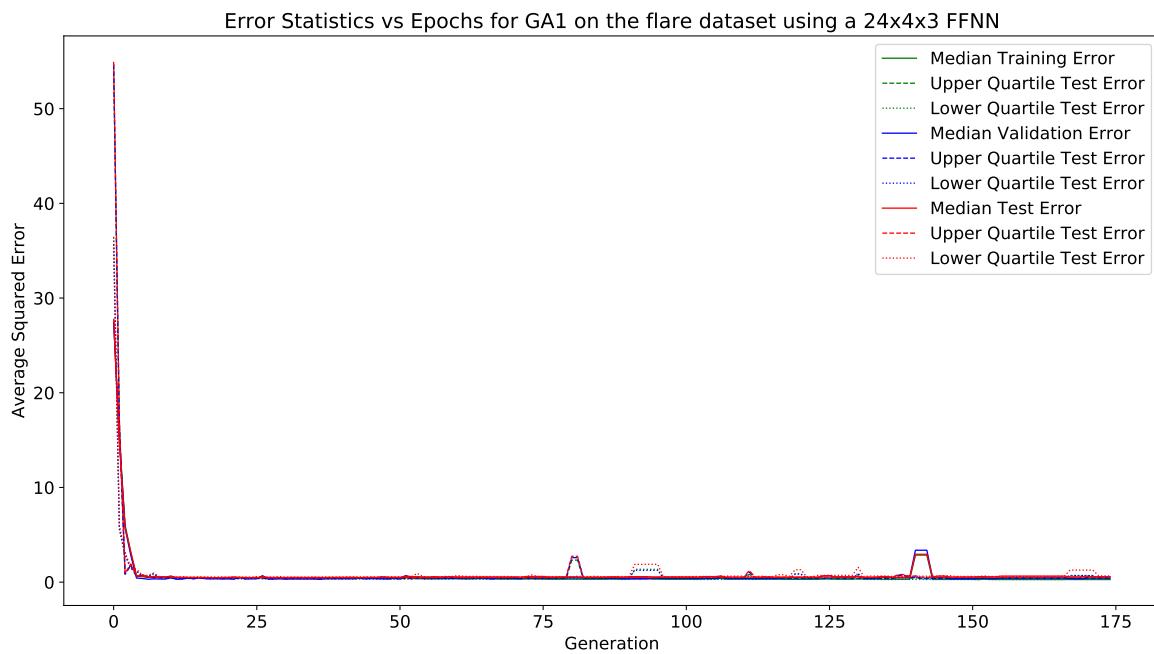
This section displays the results obtained using the flare algorithm.

#### 24 × 4 × 3 Architecture:

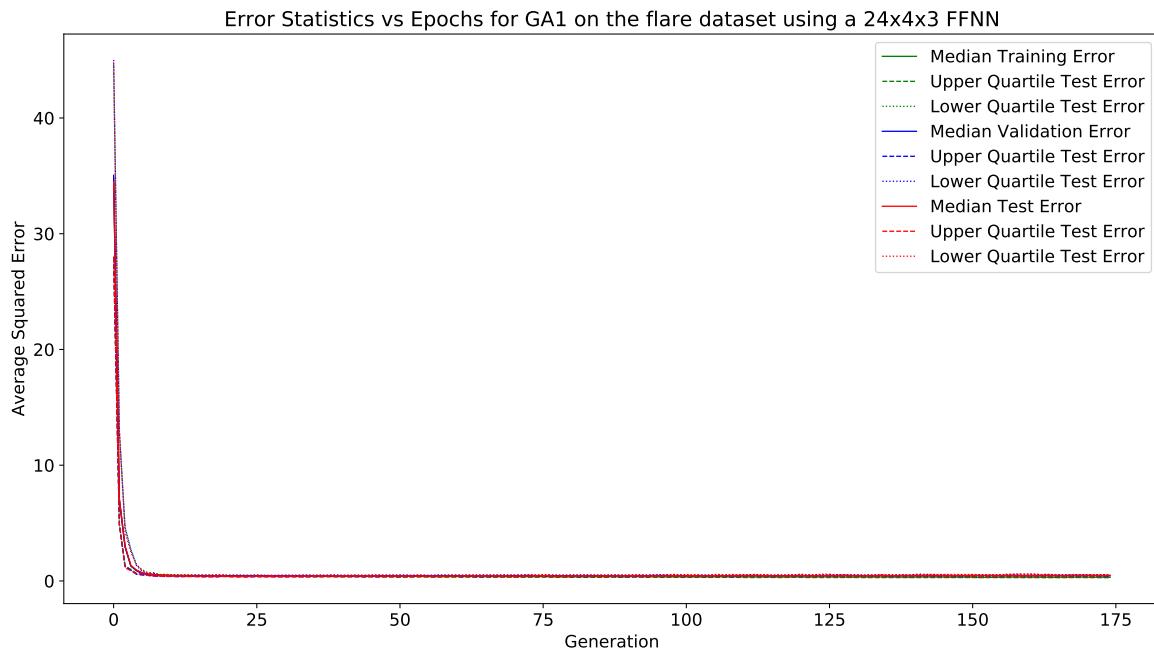
Fig. 230 shows the average and standard deviation of the test, training and validation errors. Fig. 231 shows the median and quartiles of the test, training and validation errors based on the best epoch. Finally, Fig. 232 shows the median and quartiles of the test, training and validation errors per mini-batch.



**Figure 230.** Graph of mean and standard deviation of errors vs epochs



**Figure 231.** Graph of test error vs epochs for the gradient based algorithms



**Figure 232.** Graph of test error vs epochs for the gradient based algorithms

### 16.3.9 GA2

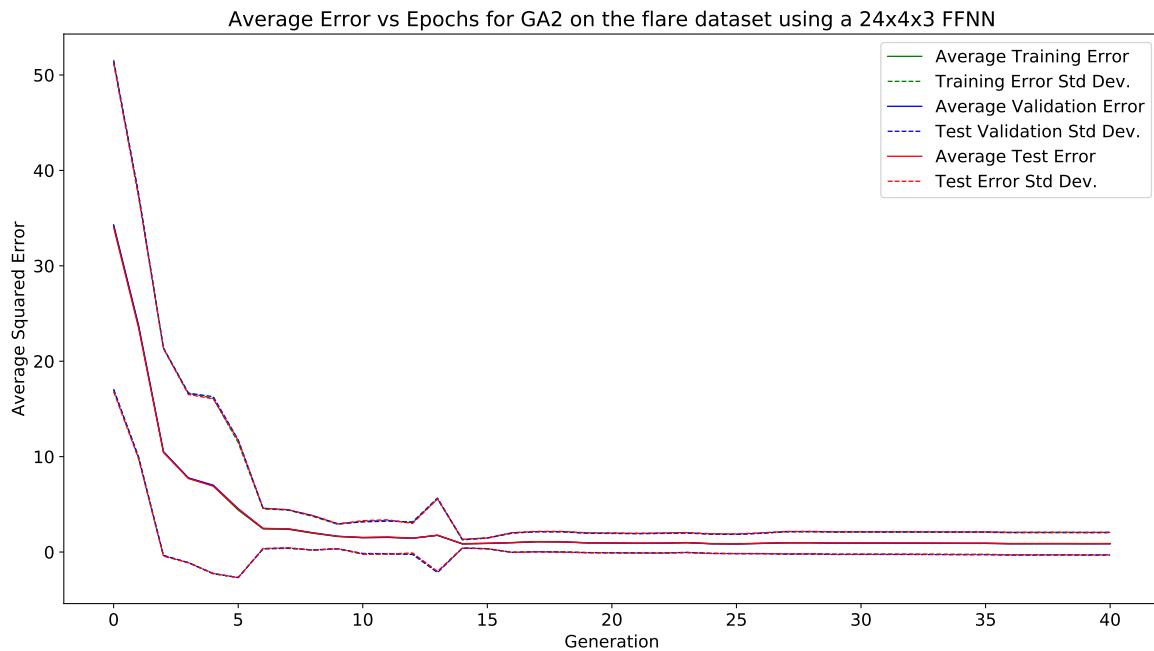
This section displays the results obtained using the GA2 algorithm.

### 16.3.10 flare

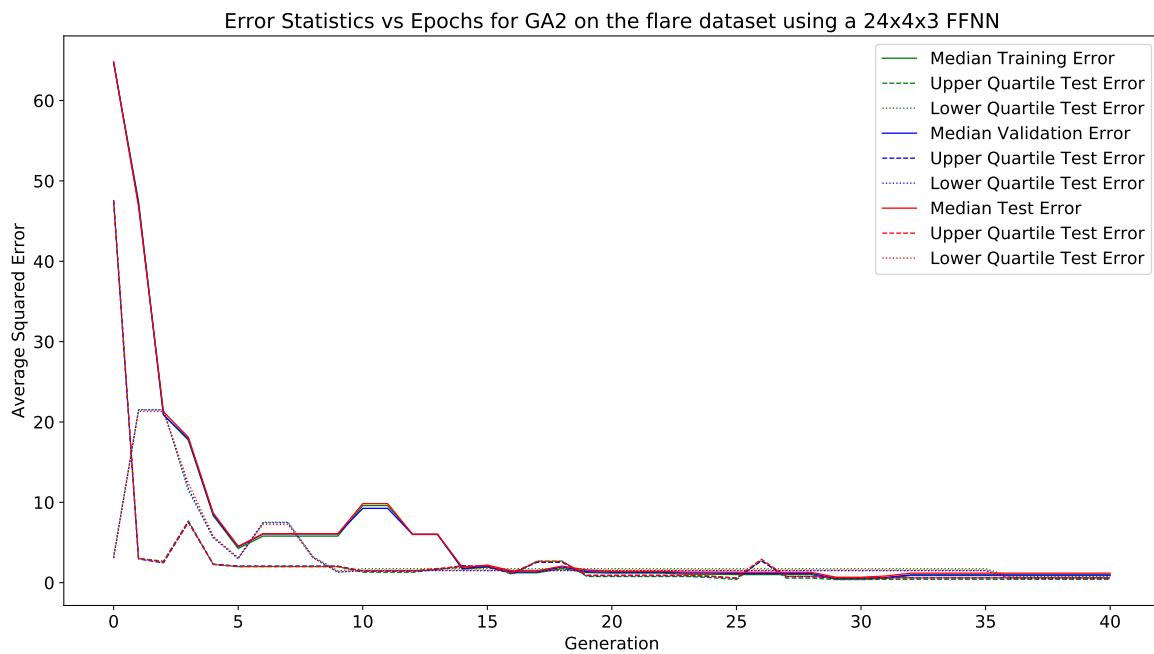
This section displays the results obtained using the flare algorithm.

#### 24 × 4 × 3 Architecture:

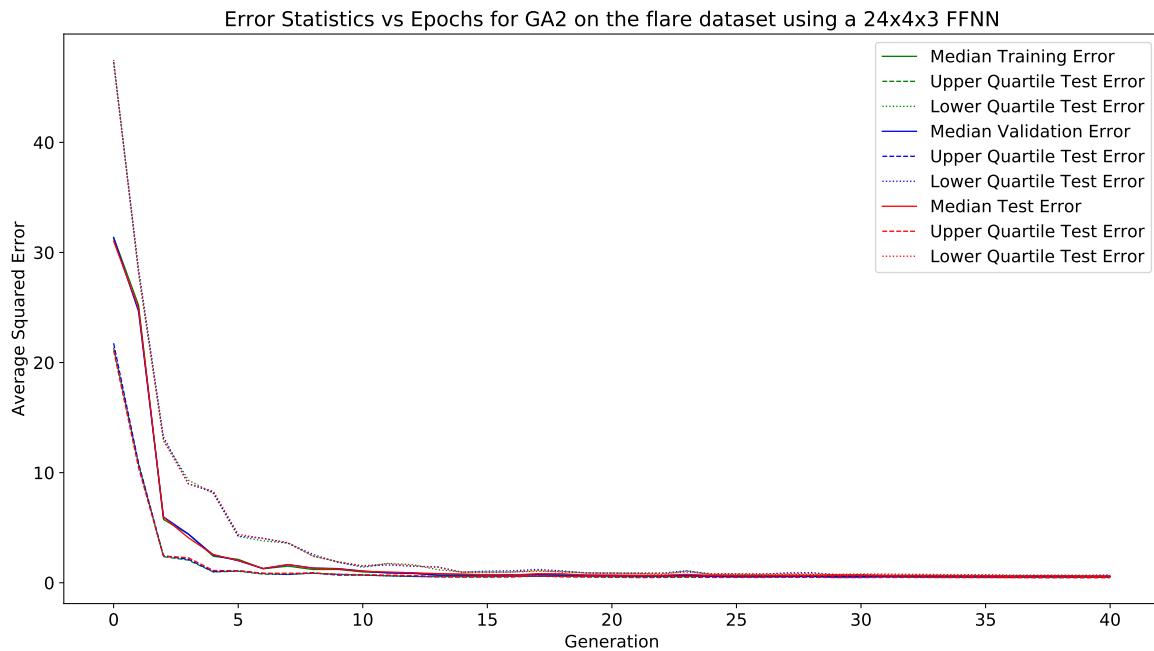
Fig. 233 shows the average and standard deviation of the test, training and validation errors. Fig. 234 shows the median and quartiles of the test, training and validation errors based on the best epoch. Finally, Fig. 235 shows the median and quartiles of the test, training and validation errors per mini-batch.



**Figure 233.** Graph of mean and standard deviation of errors vs epochs



**Figure 234.** Graph of test error vs epochs for the gradient based algorithms



**Figure 235.** Graph of test error vs epochs for the gradient based algorithms

### 16.3.11 GA3

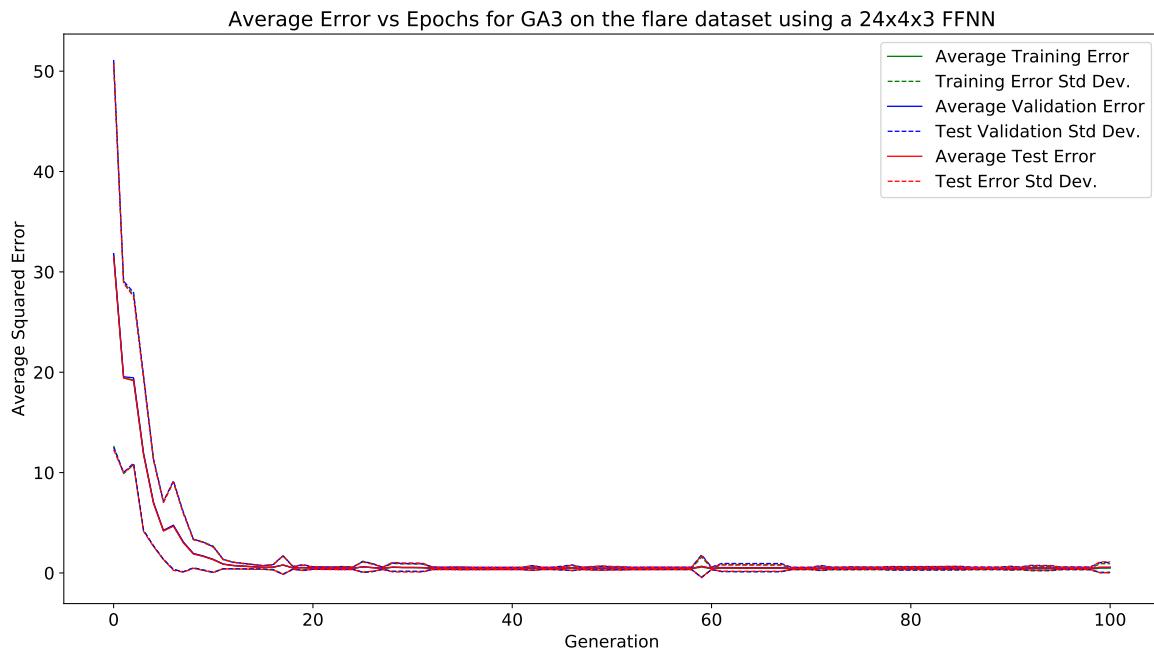
This section displays the results obtained using the GA3 algorithm.

### 16.3.12 flare

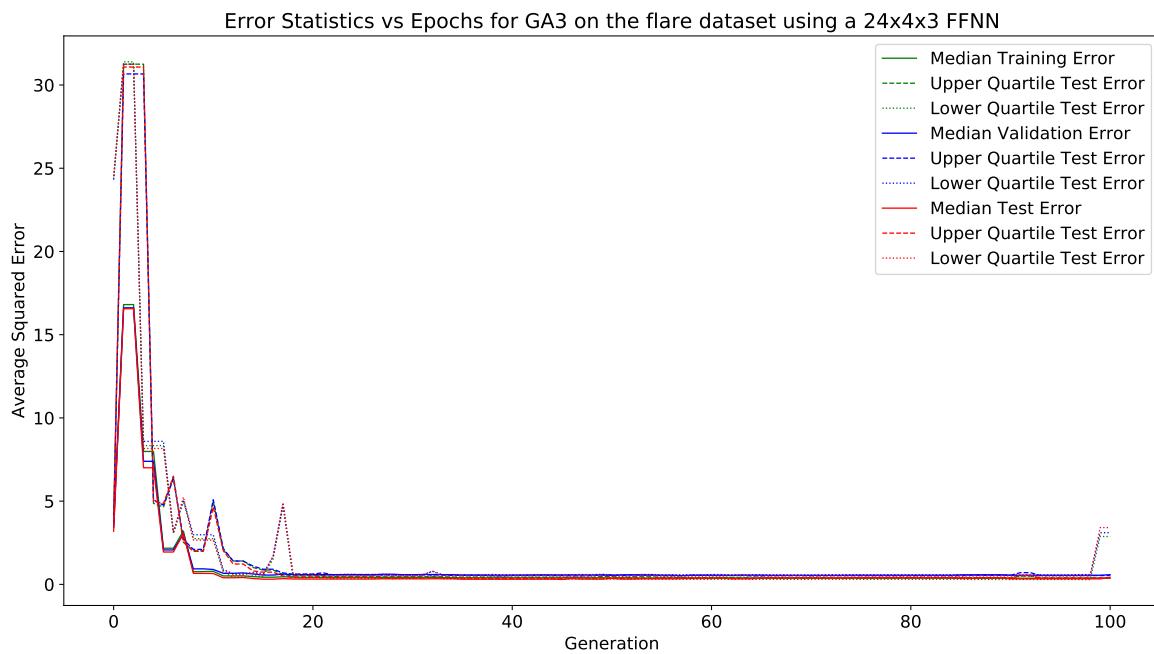
This section displays the results obtained using the flare algorithm.

#### 24 × 4 × 3 Architecture:

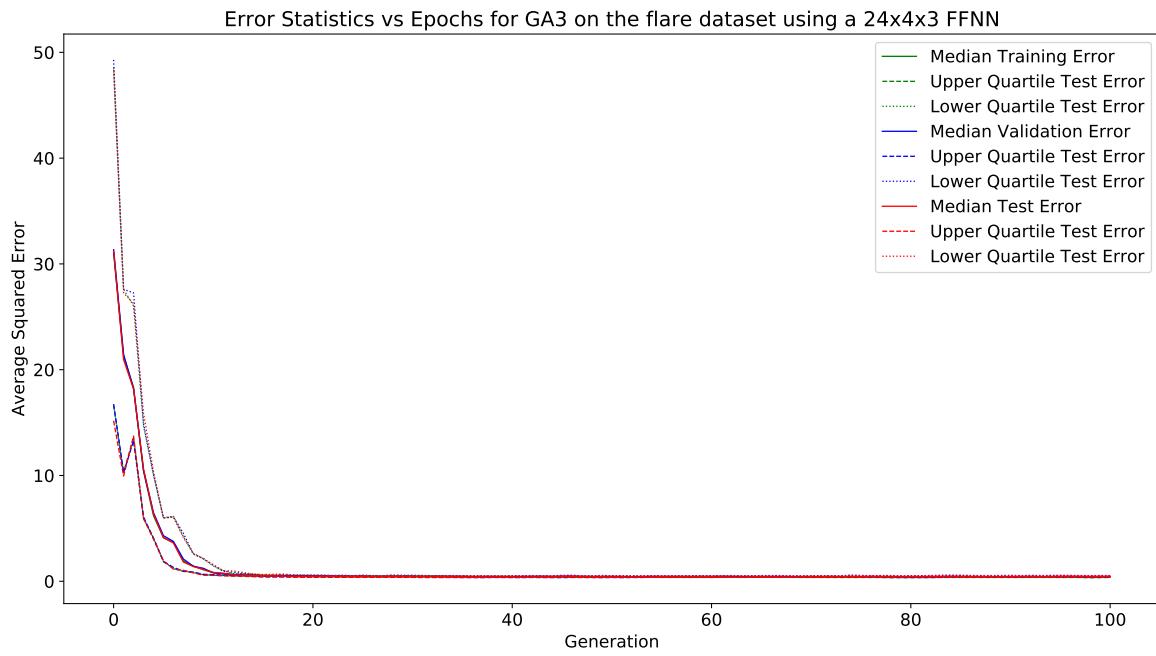
Fig. 236 shows the average and standard deviation of the test, training and validation errors. Fig. 237 shows the median and quartiles of the test, training and validation errors based on the best epoch. Finally, Fig. 238 shows the median and quartiles of the test, training and validation errors per mini-batch.



**Figure 236.** Graph of mean and standard deviation of errors vs epochs



**Figure 237.** Graph of test error vs epochs for the gradient based algorithms



**Figure 238.** Graph of test error vs epochs for the gradient based algorithms

### 16.3.13 iRPROP-

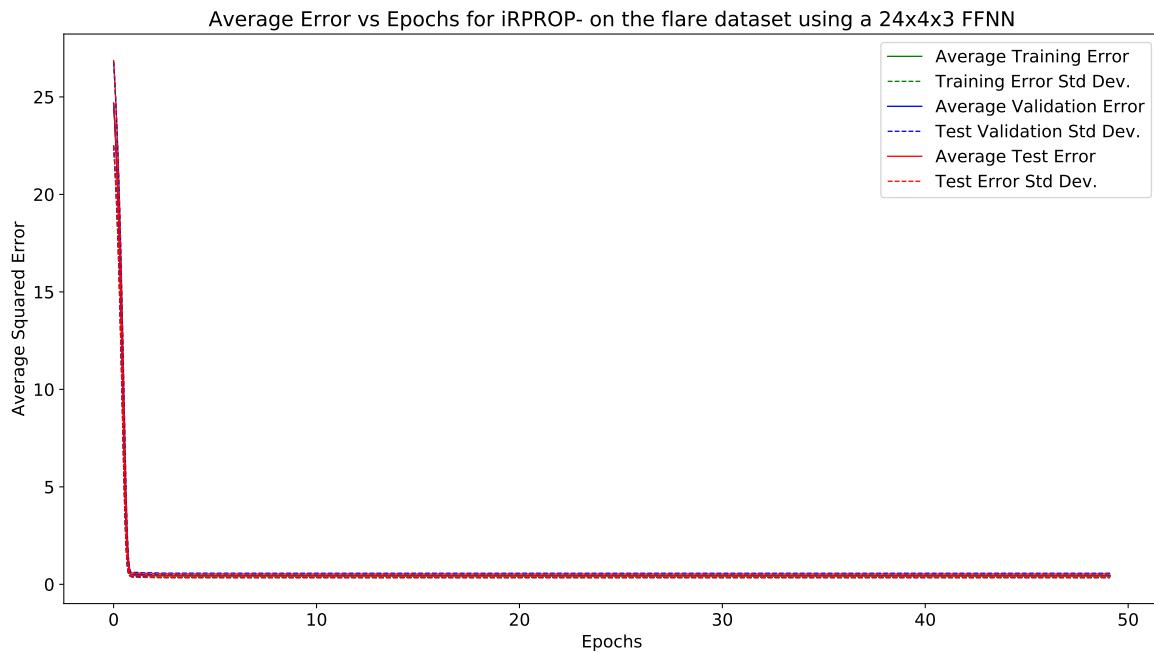
This section displays the results obtained using the iRPROP- algorithm.

### 16.3.14 flare

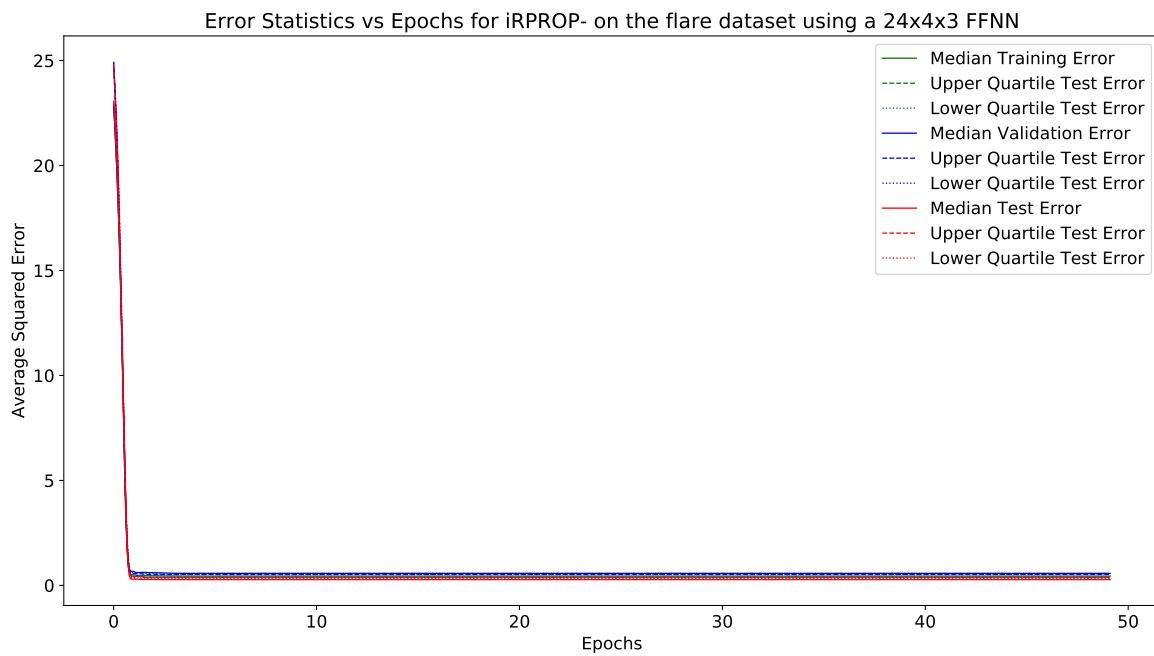
This section displays the results obtained using the flare algorithm.

#### 24 × 4 × 3 Architecture:

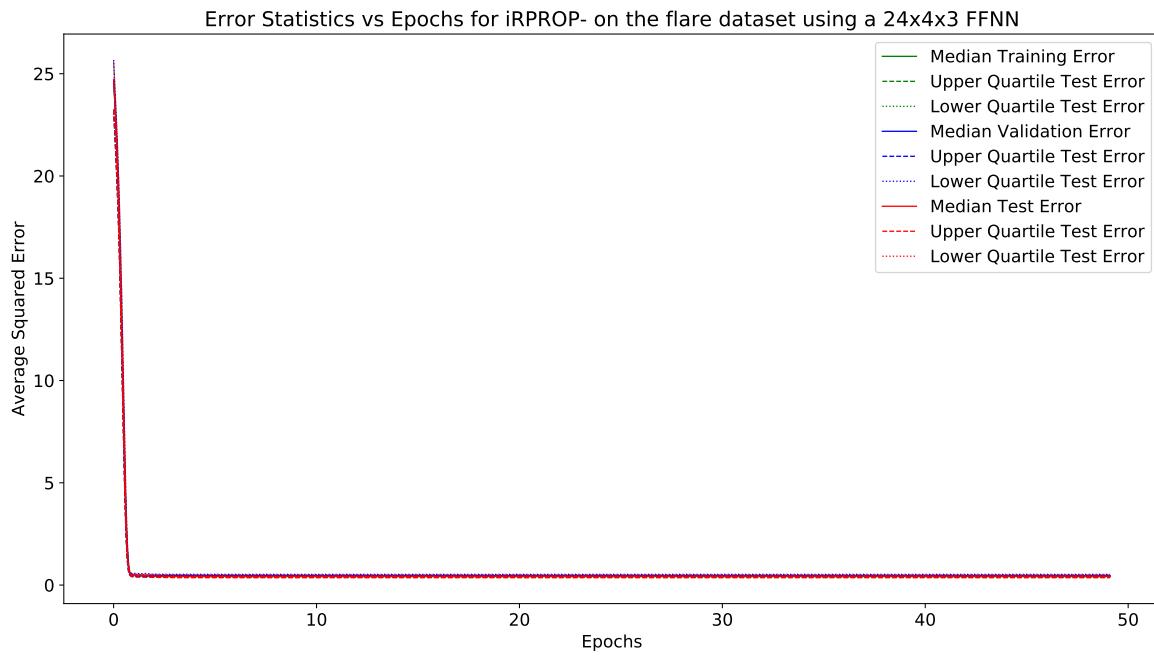
Fig. 239 shows the average and standard deviation of the test, training and validation errors. Fig. 240 shows the median and quartiles of the test, training and validation errors based on the best epoch. Finally, Fig. 241 shows the median and quartiles of the test, training and validation errors per mini-batch.



**Figure 239.** Graph of mean and standard deviation of errors vs epochs



**Figure 240.** Graph of test error vs epochs for the gradient based algorithms



**Figure 241.** Graph of test error vs epochs for the gradient based algorithms

### 16.3.15 iRPROP+

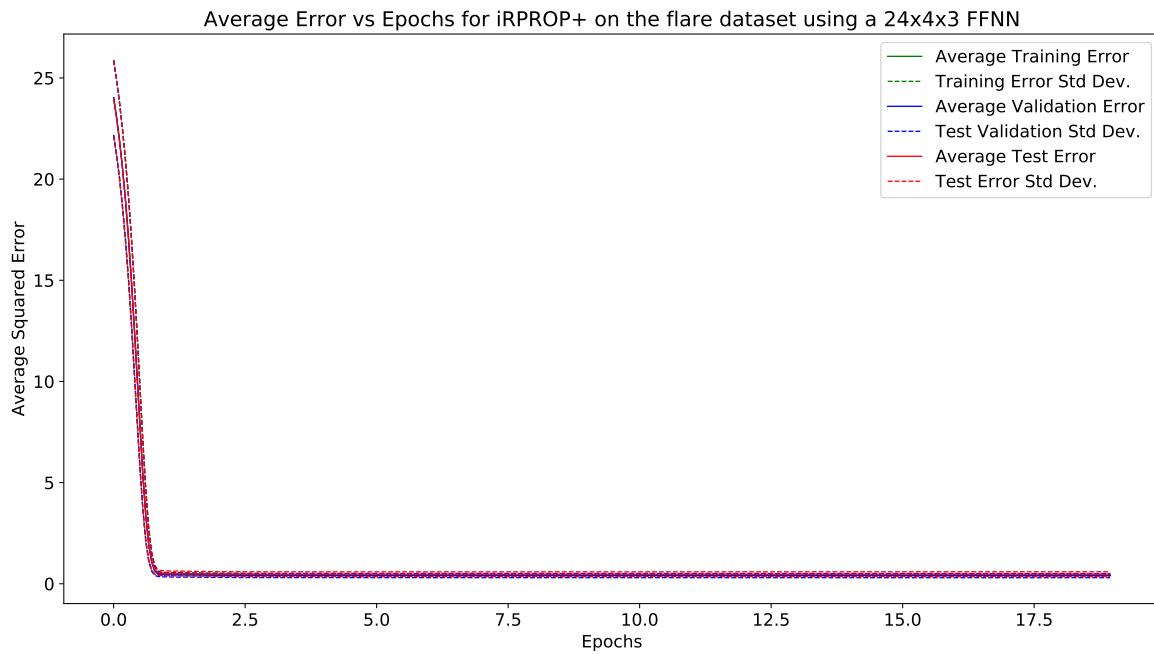
This section displays the results obtained using the iRPROP+ algorithm.

### 16.3.16 flare

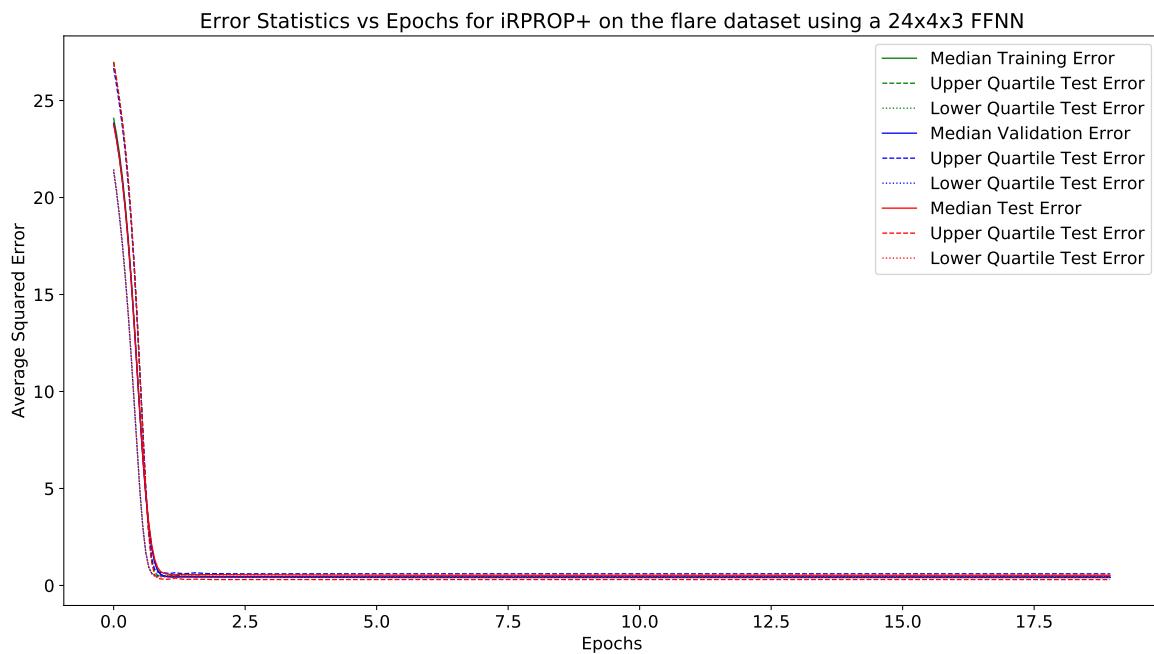
This section displays the results obtained using the flare algorithm.

#### 24 × 4 × 3 Architecture:

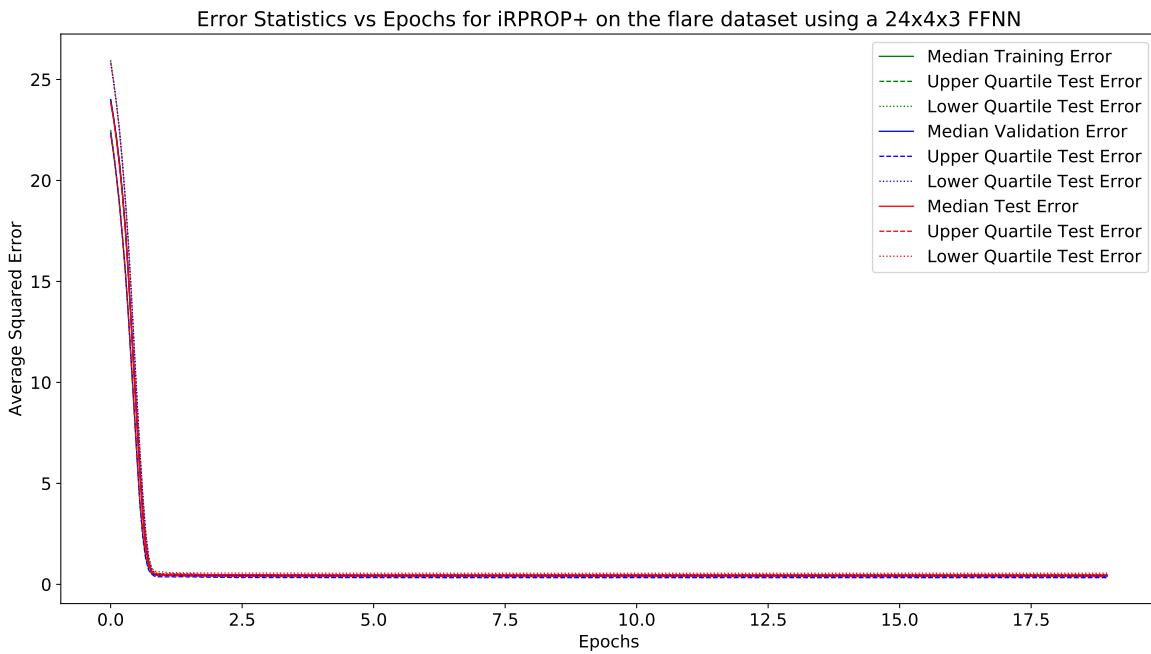
Fig. 242 shows the average and standard deviation of the test, training and validation errors. Fig. 243 shows the median and quartiles of the test, training and validation errors based on the best epoch. Finally, Fig. 244 shows the median and quartiles of the test, training and validation errors per mini-batch.



**Figure 242.** Graph of mean and standard deviation of errors vs epochs



**Figure 243.** Graph of test error vs epochs for the gradient based algorithms



**Figure 244.** Graph of test error vs epochs for the gradient based algorithms

### 16.3.17 PSO

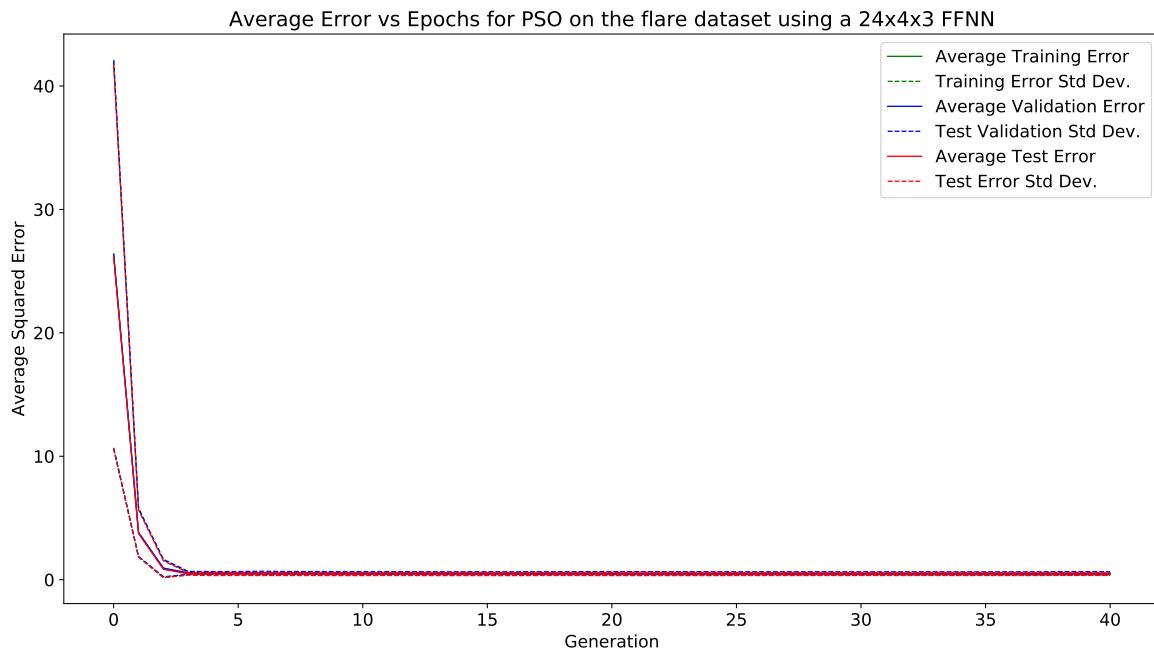
This section displays the results obtained using the PSO algorithm.

### 16.3.18 flare

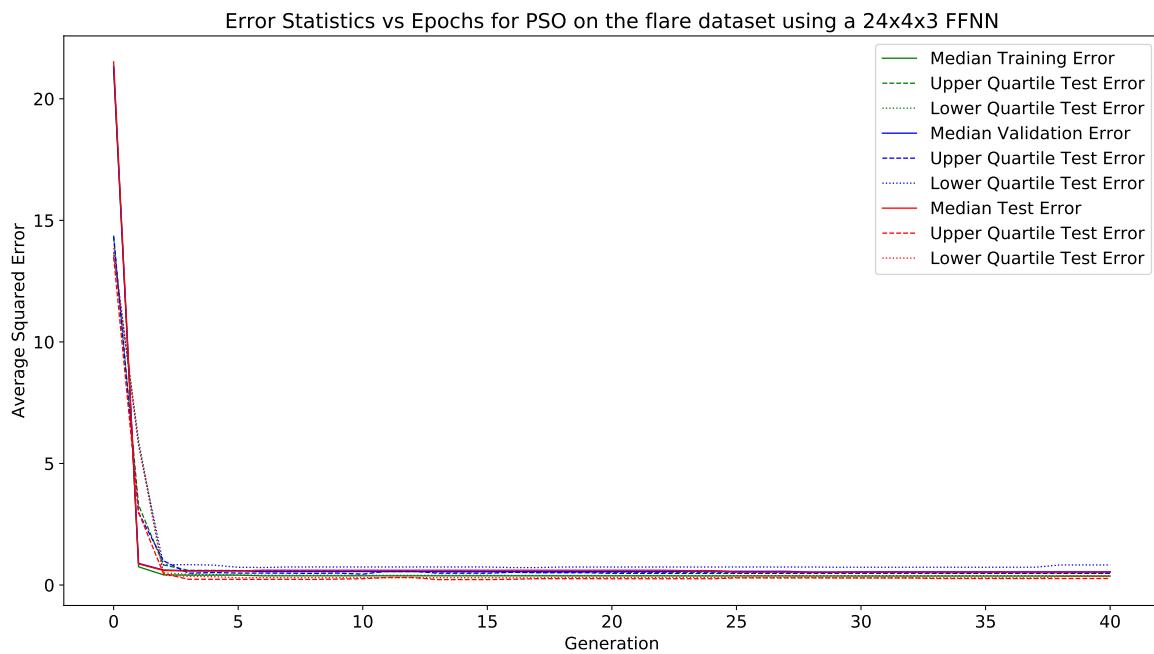
This section displays the results obtained using the flare algorithm.

#### 24 × 4 × 3 Architecture:

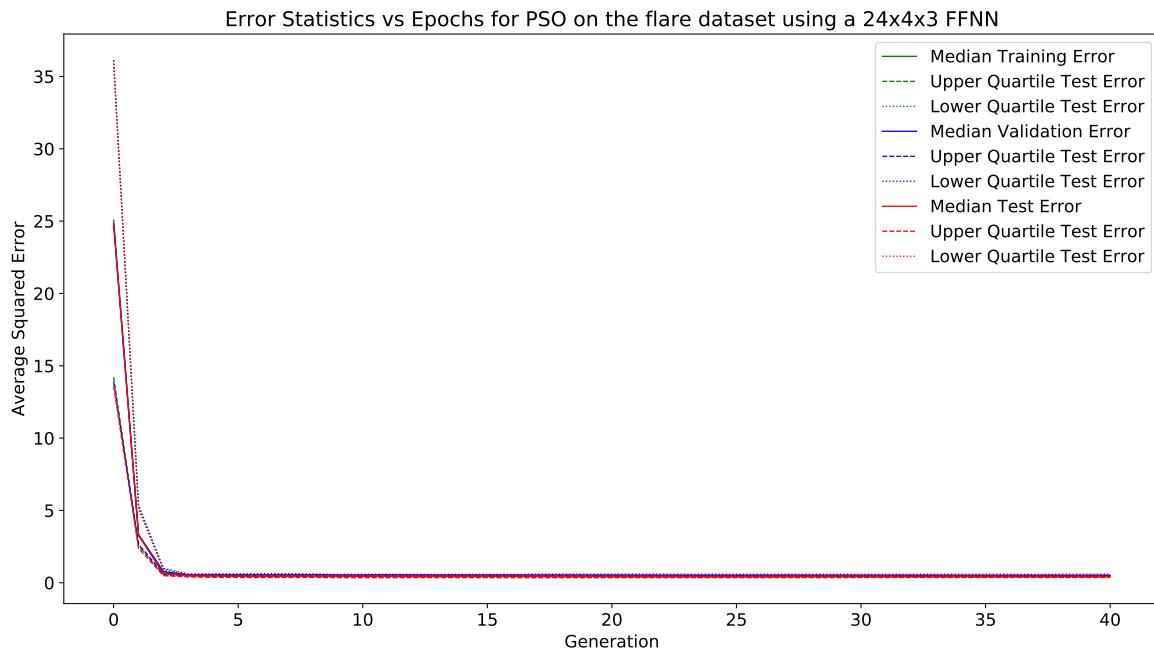
Fig. 245 shows the average and standard deviation of the test, training and validation errors. Fig. 246 shows the median and quartiles of the test, training and validation errors based on the best epoch. Finally, Fig. 247 shows the median and quartiles of the test, training and validation errors per mini-batch.



**Figure 245.** Graph of mean and standard deviation of errors vs epochs



**Figure 246.** Graph of test error vs epochs for the gradient based algorithms



**Figure 247.** Graph of test error vs epochs for the gradient based algorithms

### 16.3.19 QuickProp

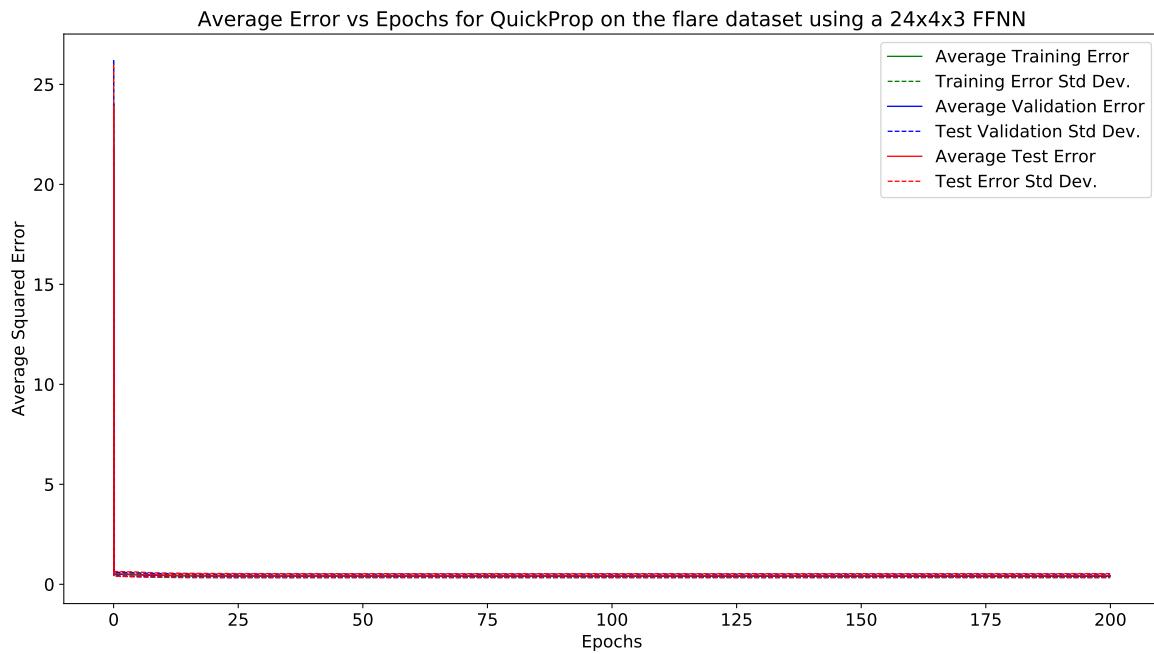
This section displays the results obtained using the QuickProp algorithm.

### 16.3.20 flare

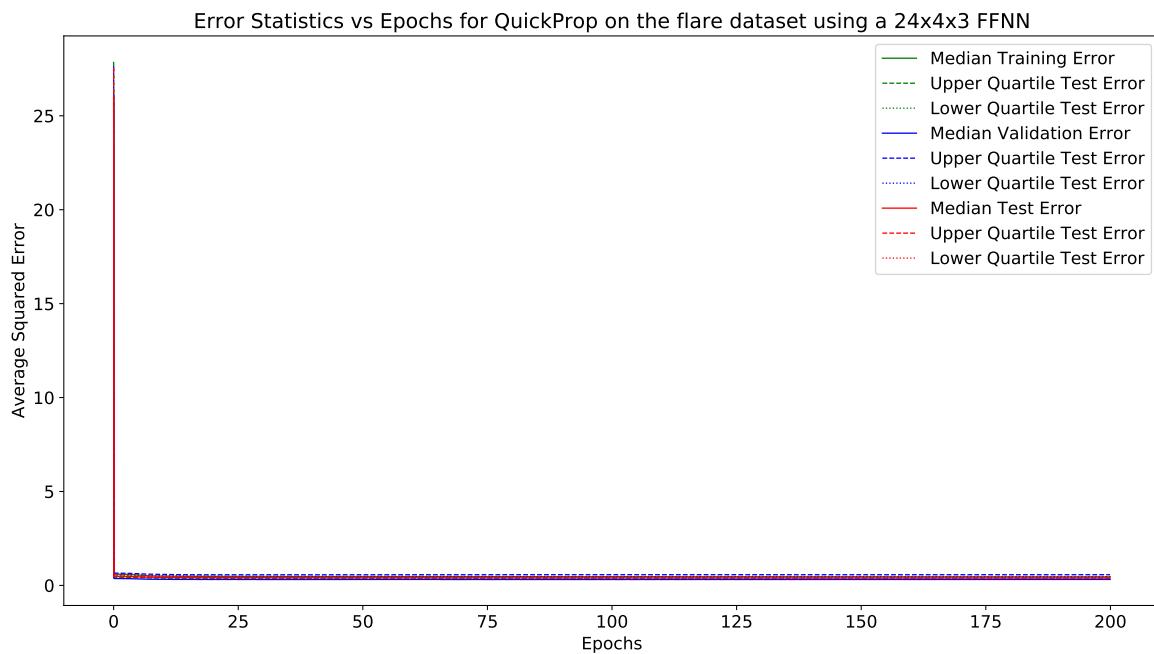
This section displays the results obtained using the flare algorithm.

#### 24 × 4 × 3 Architecture:

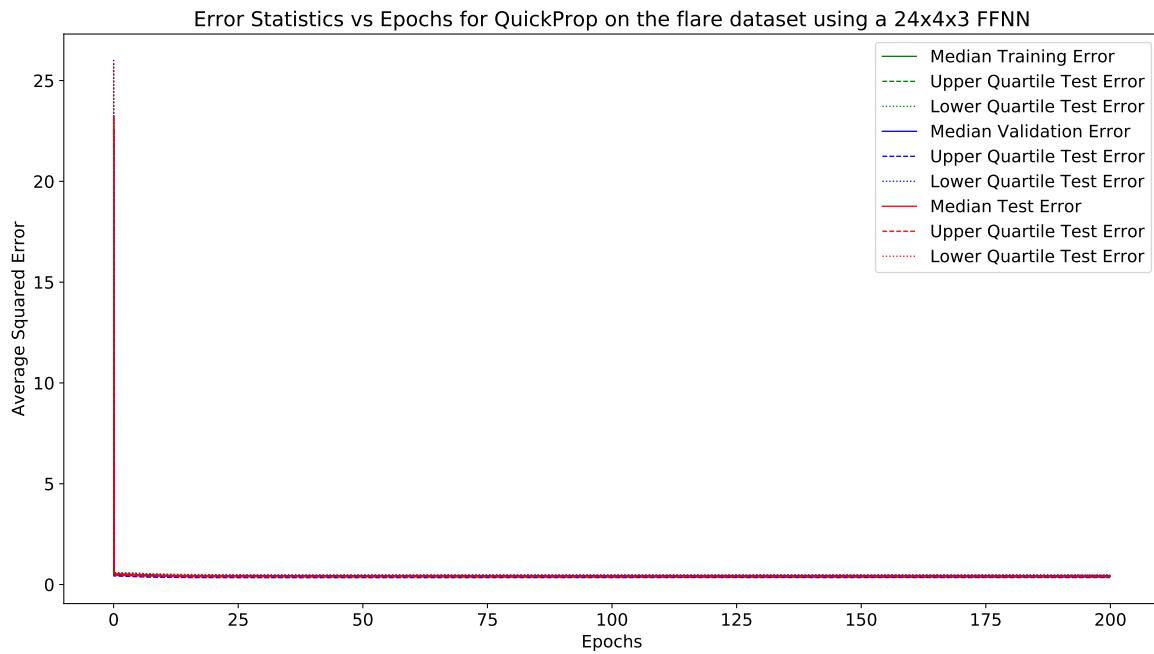
Fig. 248 shows the average and standard deviation of the test, training and validation errors. Fig. 249 shows the median and quartiles of the test, training and validation errors based on the best epoch. Finally, Fig. 250 shows the median and quartiles of the test, training and validation errors per mini-batch.



**Figure 248.** Graph of mean and standard deviation of errors vs epochs



**Figure 249.** Graph of test error vs epochs for the gradient based algorithms



**Figure 250.** Graph of test error vs epochs for the gradient based algorithms

### 16.3.21 RPROP-

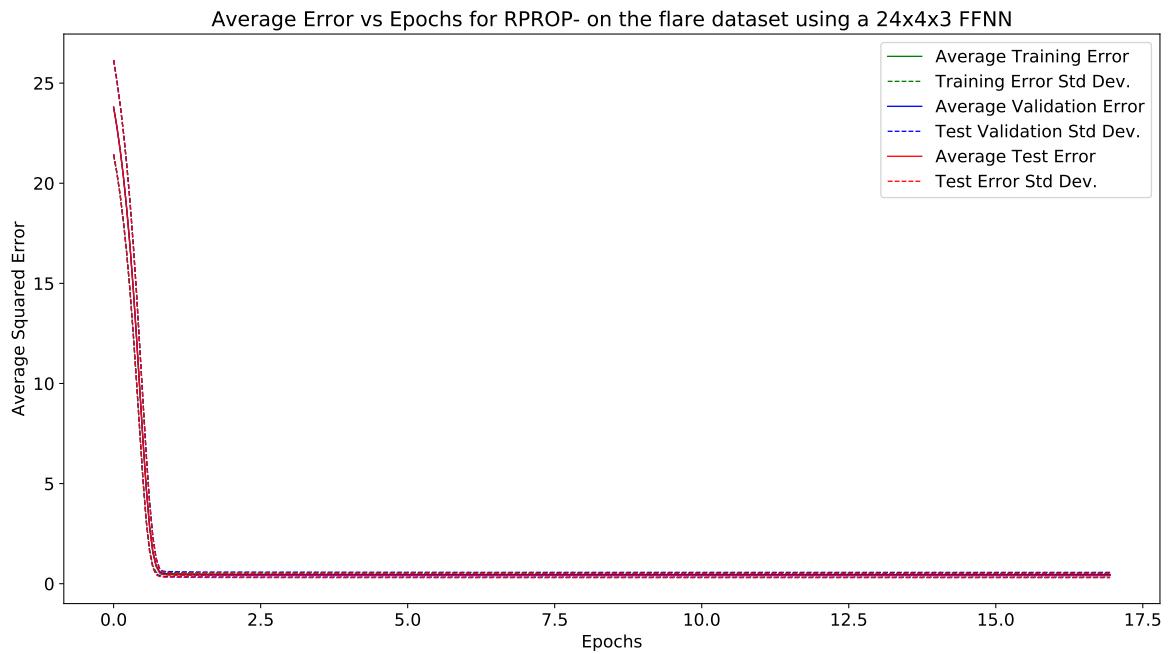
This section displays the results obtained using the RPROP- algorithm.

### 16.3.22 flare

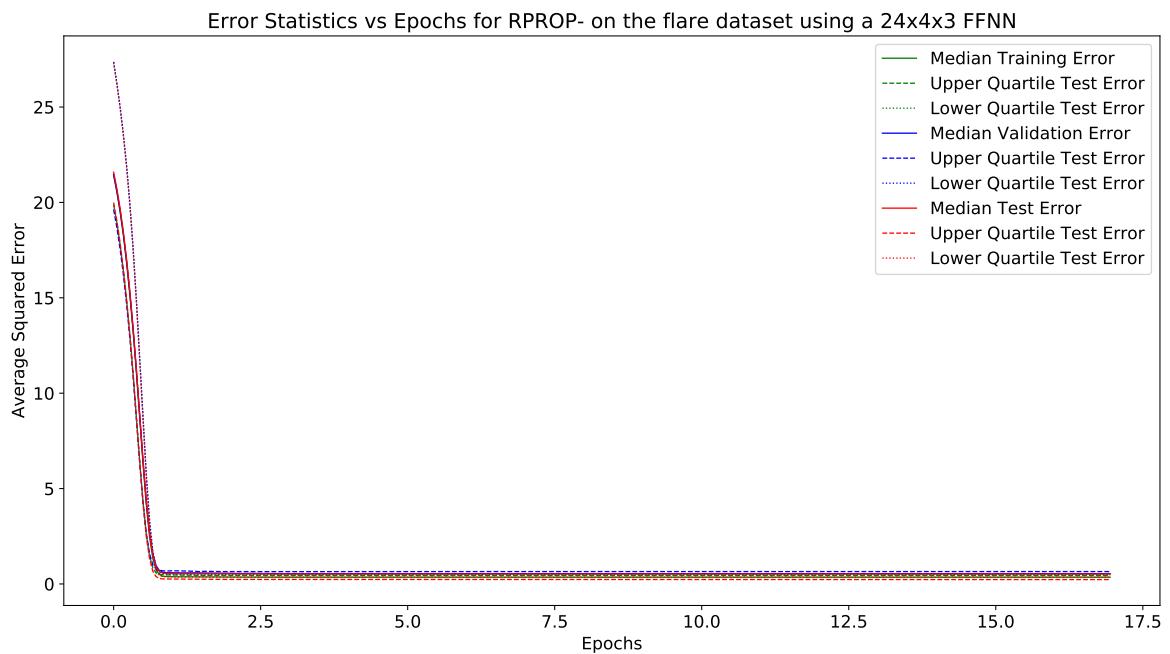
This section displays the results obtained using the flare algorithm.

#### 24 × 4 × 3 Architecture:

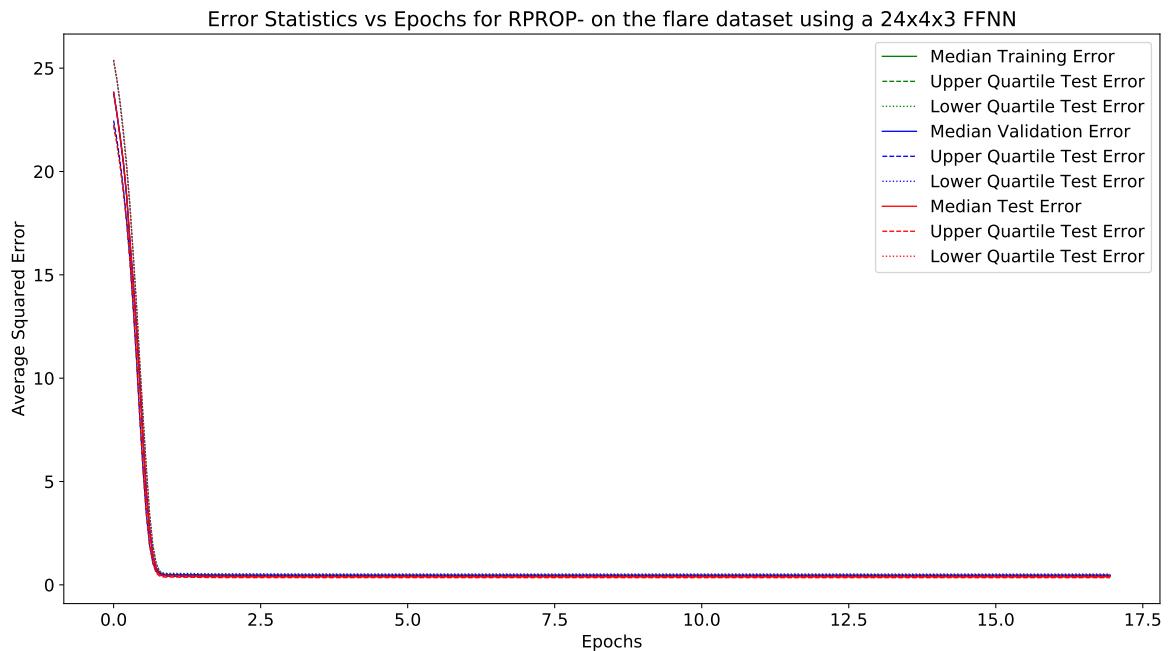
Fig. 251 shows the average and standard deviation of the test, training and validation errors. Fig. 252 shows the median and quartiles of the test, training and validation errors based on the best epoch. Finally, Fig. 253 shows the median and quartiles of the test, training and validation errors per mini-batch.



**Figure 251.** Graph of mean and standard deviation of errors vs epochs



**Figure 252.** Graph of test error vs epochs for the gradient based algorithms



**Figure 253.** Graph of test error vs epochs for the gradient based algorithms

### 16.3.23 RPROP+

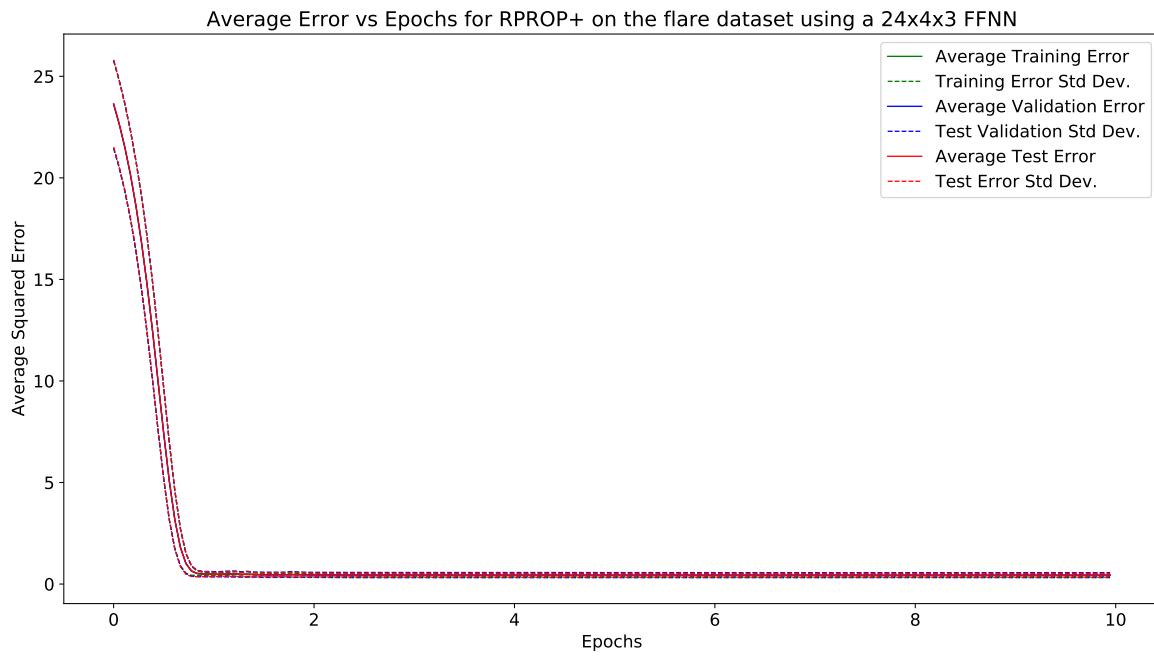
This section displays the results obtained using the RPROP+ algorithm.

### 16.3.24 flare

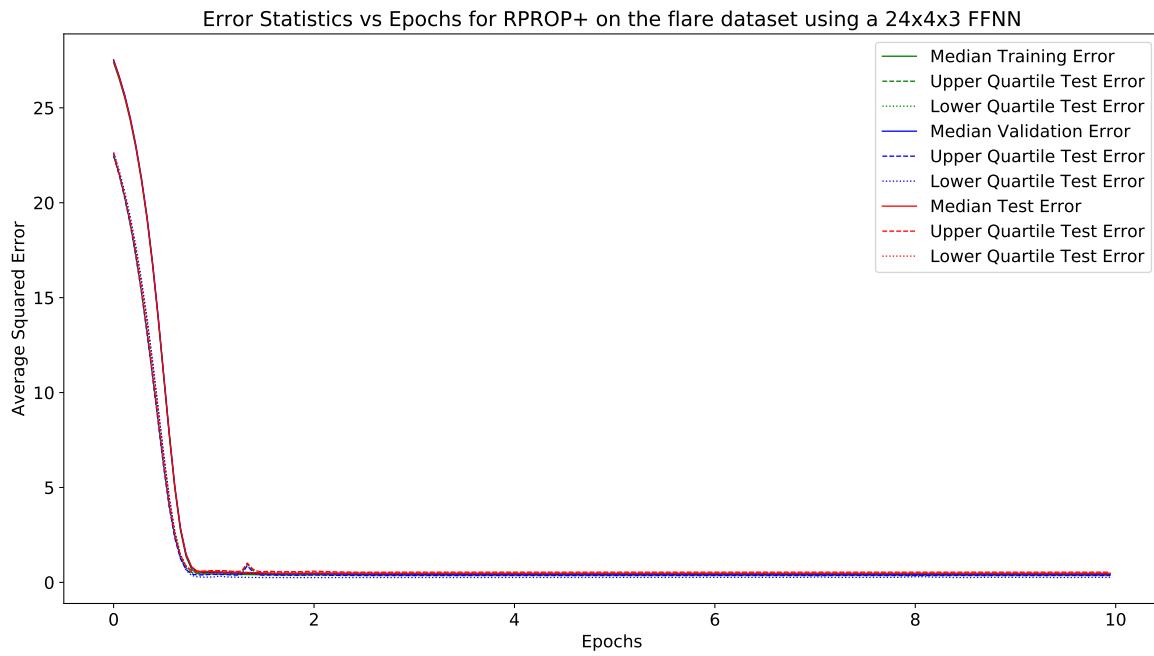
This section displays the results obtained using the flare algorithm.

#### 24 × 4 × 3 Architecture:

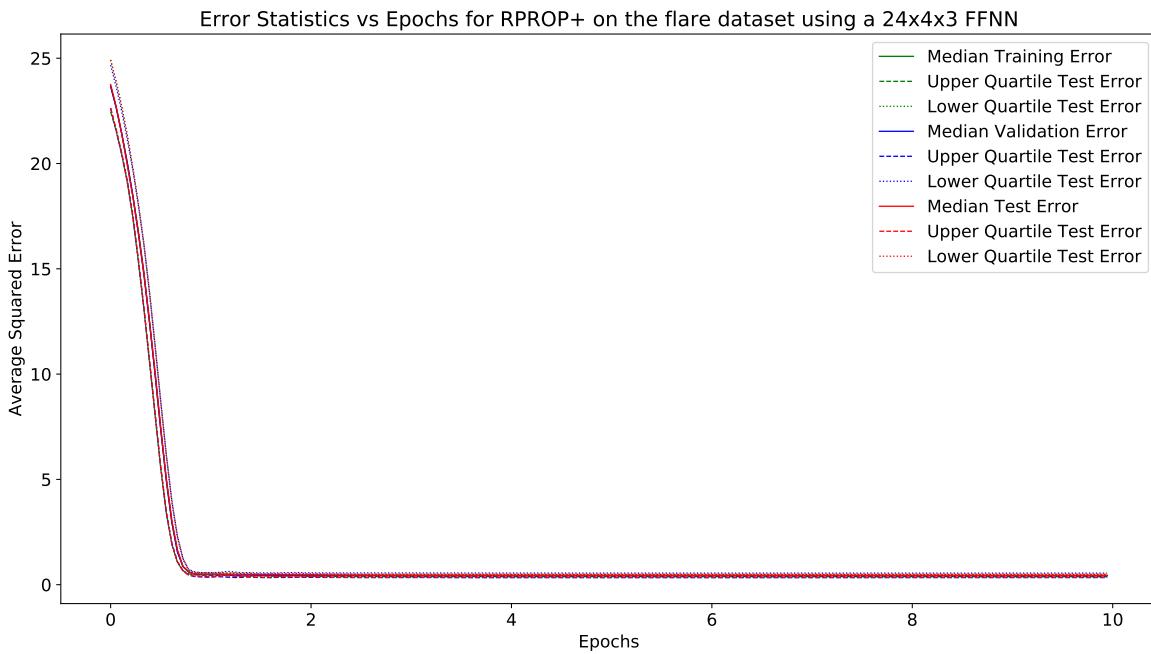
Fig. 254 shows the average and standard deviation of the test, training and validation errors. Fig. 255 shows the median and quartiles of the test, training and validation errors based on the best epoch. Finally, Fig. 256 shows the median and quartiles of the test, training and validation errors per mini-batch.



**Figure 254.** Graph of mean and standard deviation of errors vs epochs



**Figure 255.** Graph of test error vs epochs for the gradient based algorithms



**Figure 256.** Graph of test error vs epochs for the gradient based algorithms

## 16.4 gene

This section displays the results obtained in the gene dataset.

### 16.4.1 ADAGRAD

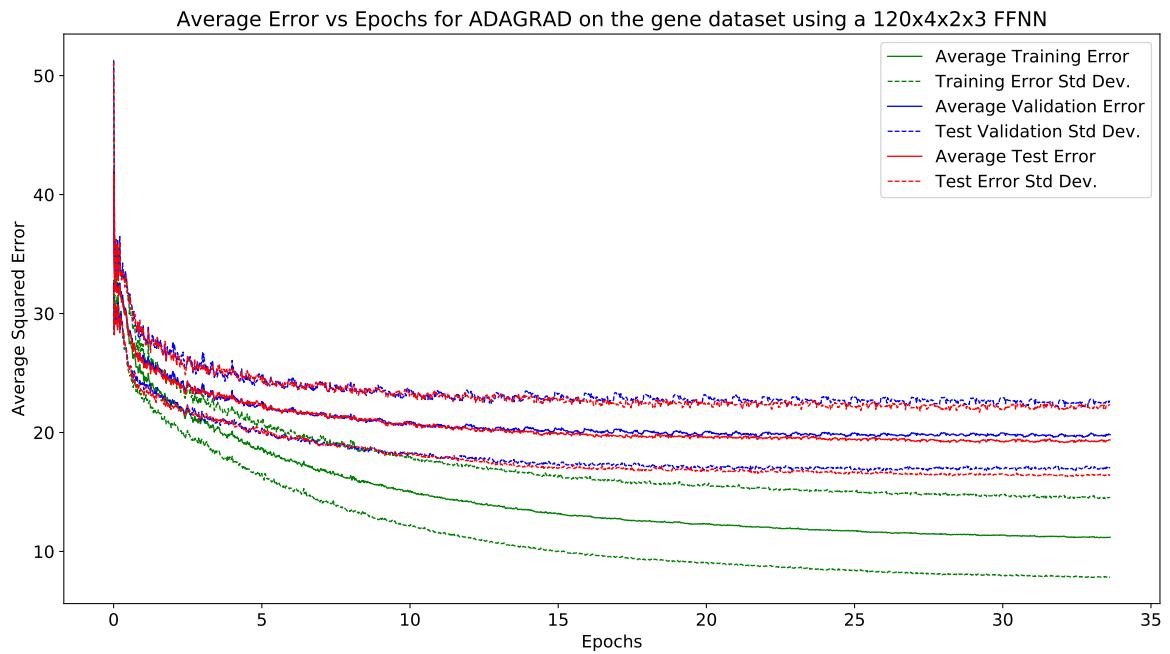
This section displays the results obtained using the ADAGRAD algorithm.

### 16.4.2 gene

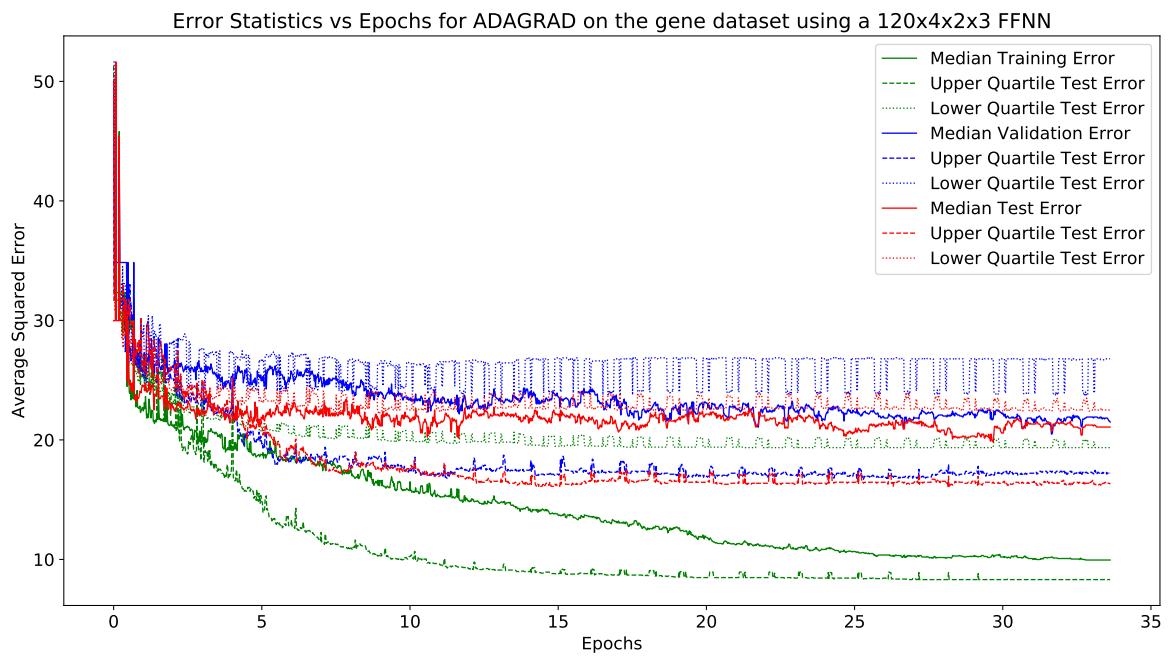
This section displays the results obtained using the gene algorithm.

#### 120 × 4 × 2 × 3 Architecture:

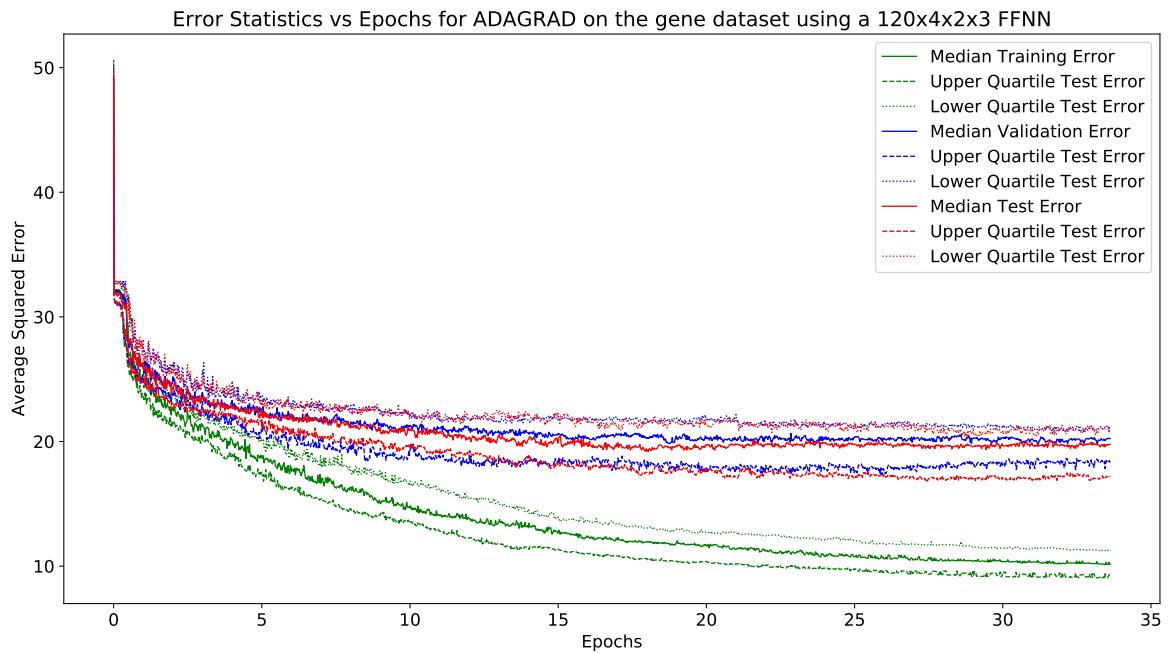
Fig. 257 shows the average and standard deviation of the test, training and validation errors. Fig. 258 shows the median and quartiles of the test, training and validation errors based on the best epoch. Finally, Fig. 259 shows the median and quartiles of the test, training and validation errors per mini-batch.



**Figure 257.** Graph of mean and standard deviation of errors vs epochs



**Figure 258.** Graph of test error vs epochs for the gradient based algorithms



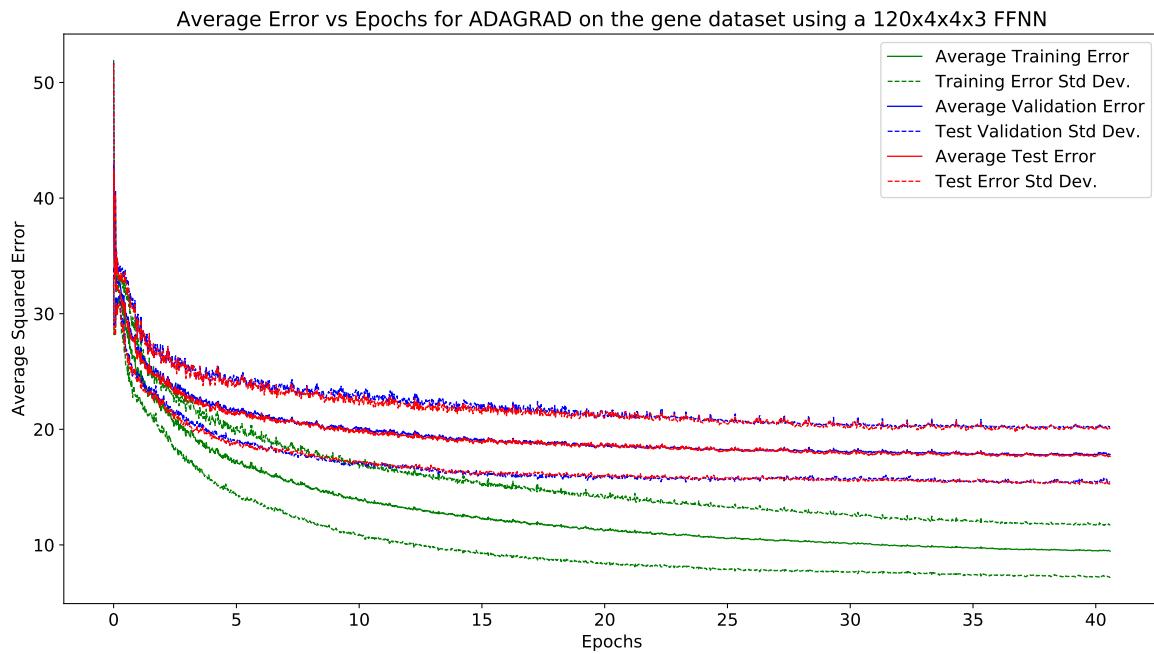
**Figure 259.** Graph of test error vs epochs for the gradient based algorithms

### 16.4.3 gene

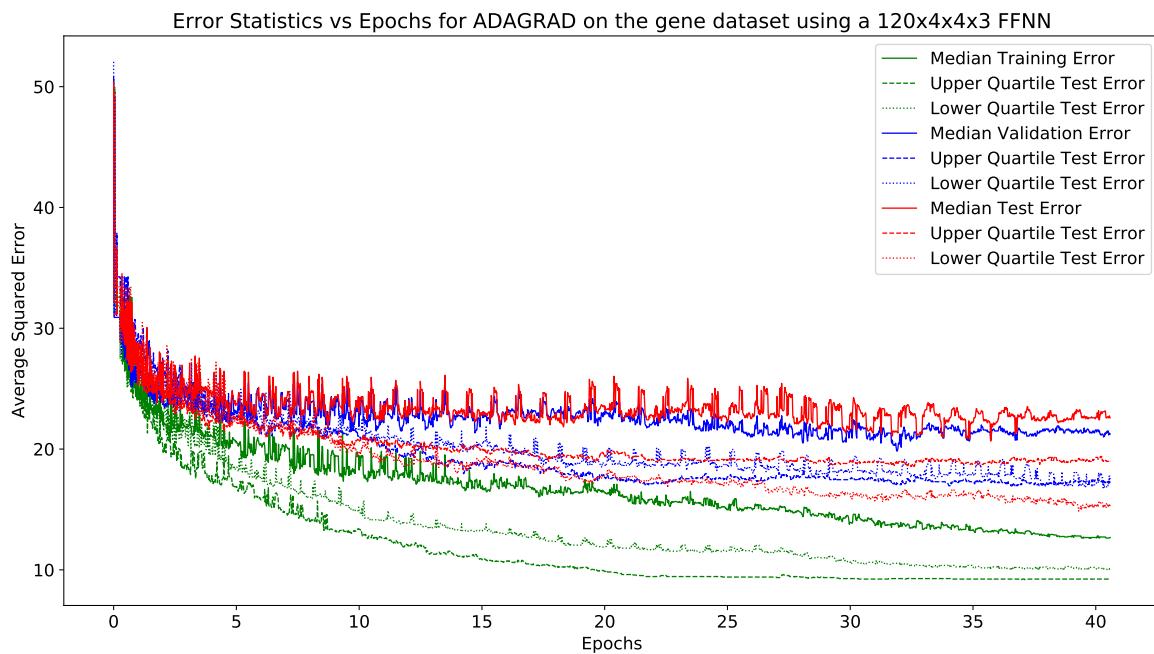
This section displays the results obtained using the gene algorithm.

#### 120 × 4 × 4 × 3 Architecture:

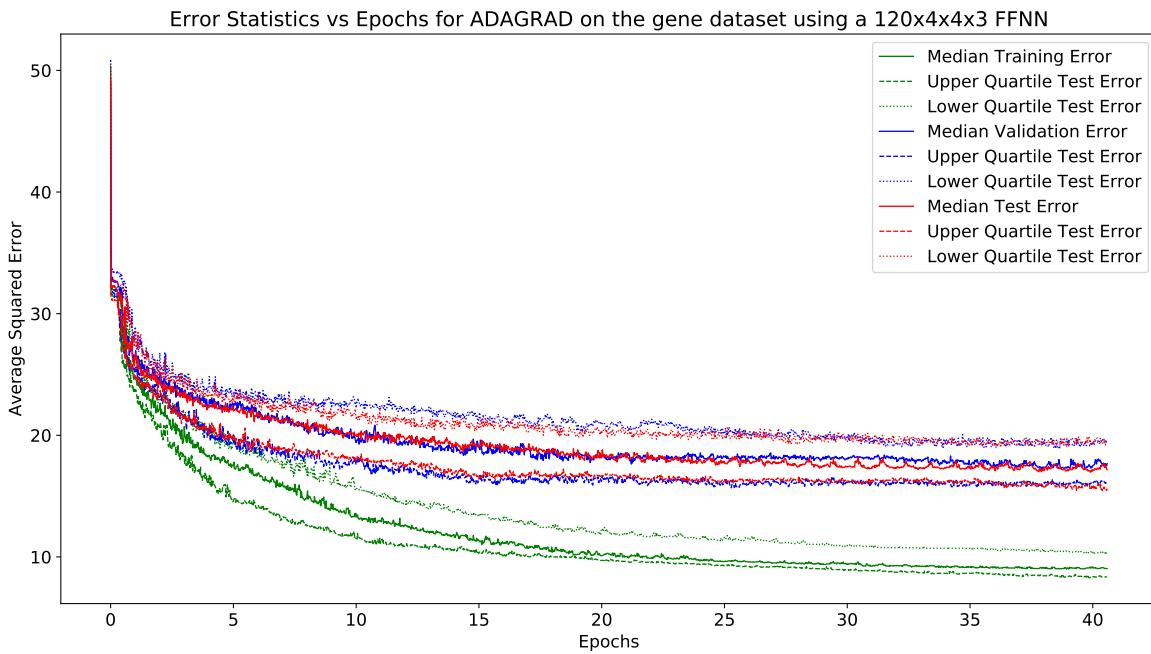
Fig. 260 shows the average and standard deviation of the test, training and validation errors. Fig. 261 shows the median and quartiles of the test, training and validation errors based on the best epoch. Finally, Fig. 262 shows the median and quartiles of the test, training and validation errors per mini-batch.



**Figure 260.** Graph of mean and standard deviation of errors vs epochs



**Figure 261.** Graph of test error vs epochs for the gradient based algorithms



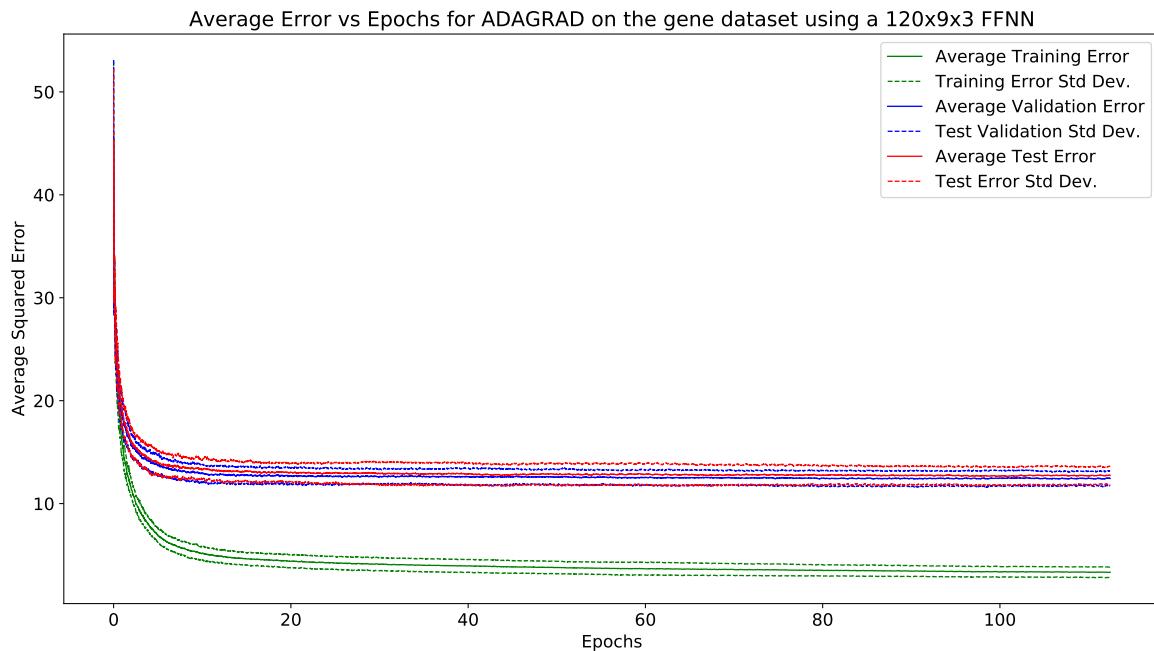
**Figure 262.** Graph of test error vs epochs for the gradient based algorithms

#### 16.4.4 gene

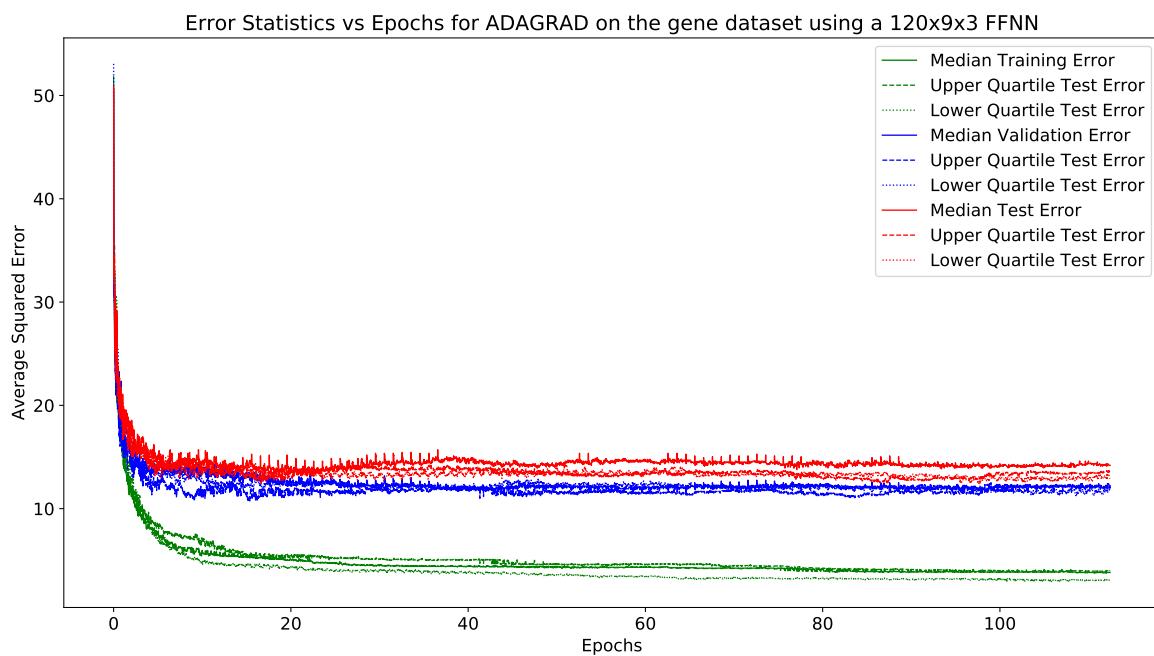
This section displays the results obtained using the gene algorithm.

##### 120 × 9 × 3 Architecture:

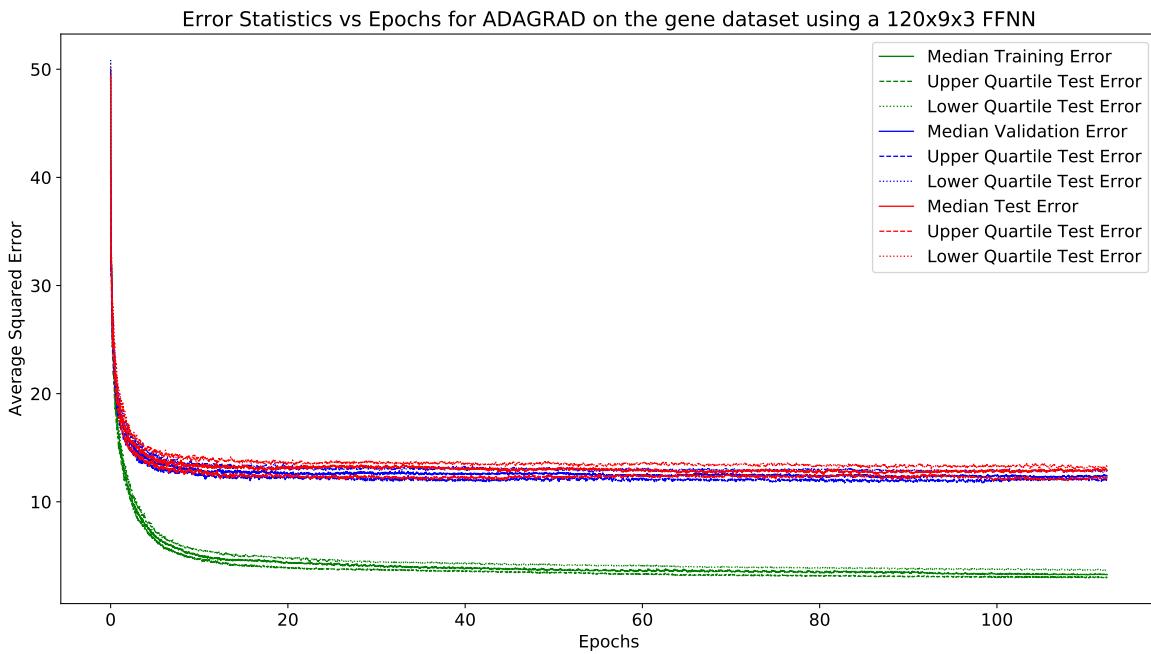
Fig. 263 shows the average and standard deviation of the test, training and validation errors. Fig. 264 shows the median and quartiles of the test, training and validation errors based on the best epoch. Finally, Fig. 265 shows the median and quartiles of the test, training and validation errors per mini-batch.



**Figure 263.** Graph of mean and standard deviation of errors vs epochs



**Figure 264.** Graph of test error vs epochs for the gradient based algorithms



**Figure 265.** Graph of test error vs epochs for the gradient based algorithms

#### 16.4.5 Backprop with Momentum

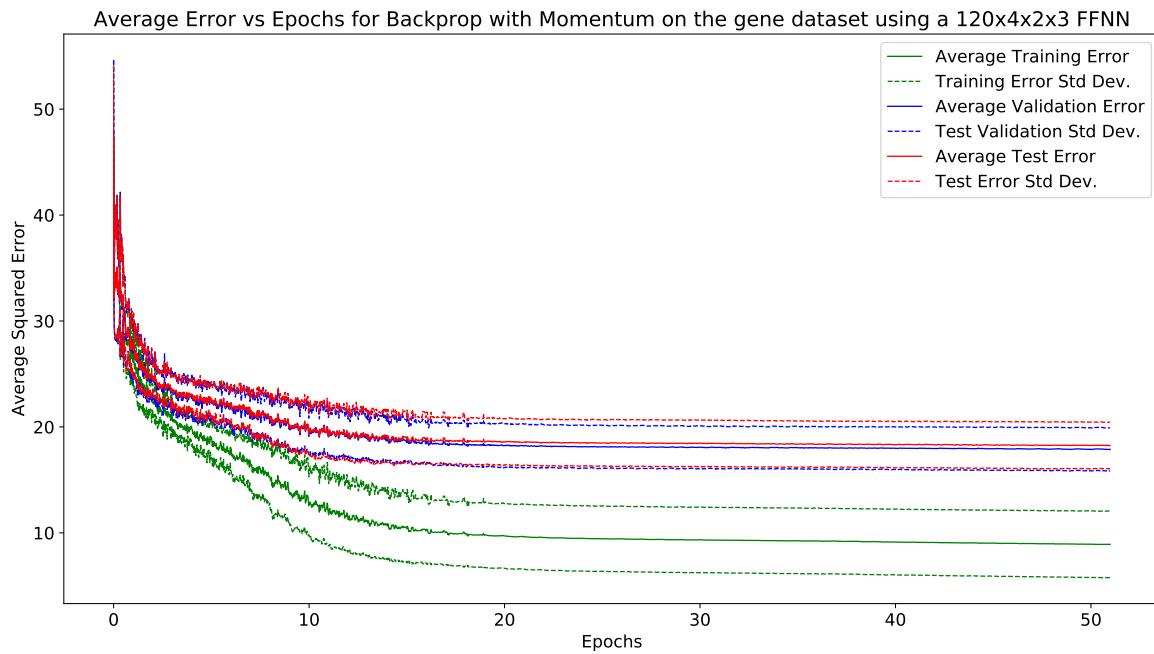
This section displays the results obtained using the Backprop with Momentum algorithm.

#### 16.4.6 gene

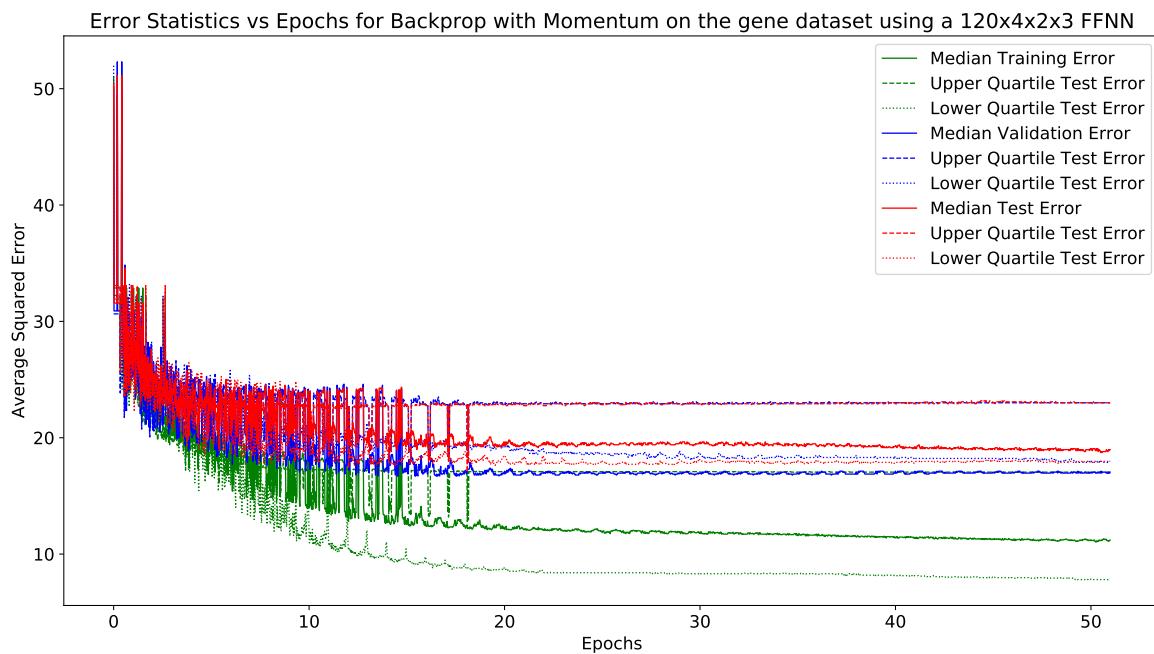
This section displays the results obtained using the gene algorithm.

##### 120 × 4 × 2 × 3 Architecture:

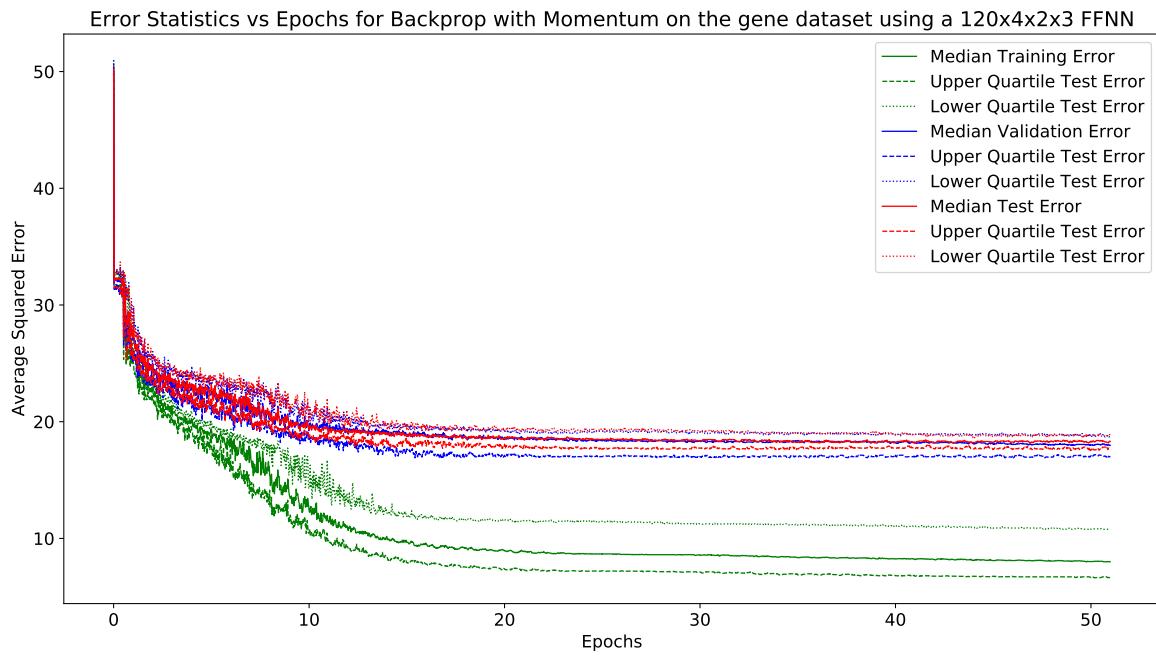
Fig. 266 shows the average and standard deviation of the test, training and validation errors. Fig. 267 shows the median and quartiles of the test, training and validation errors based on the best epoch. Finally, Fig. 268 shows the median and quartiles of the test, training and validation errors per mini-batch.



**Figure 266.** Graph of mean and standard deviation of errors vs epochs



**Figure 267.** Graph of test error vs epochs for the gradient based algorithms



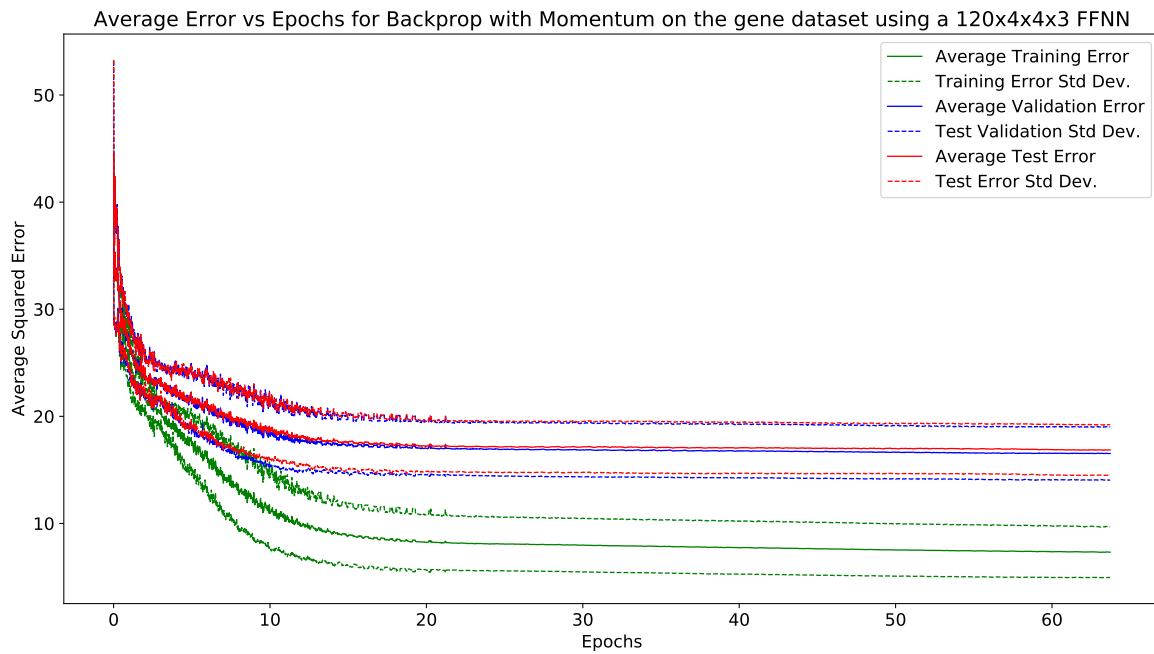
**Figure 268.** Graph of test error vs epochs for the gradient based algorithms

#### 16.4.7 gene

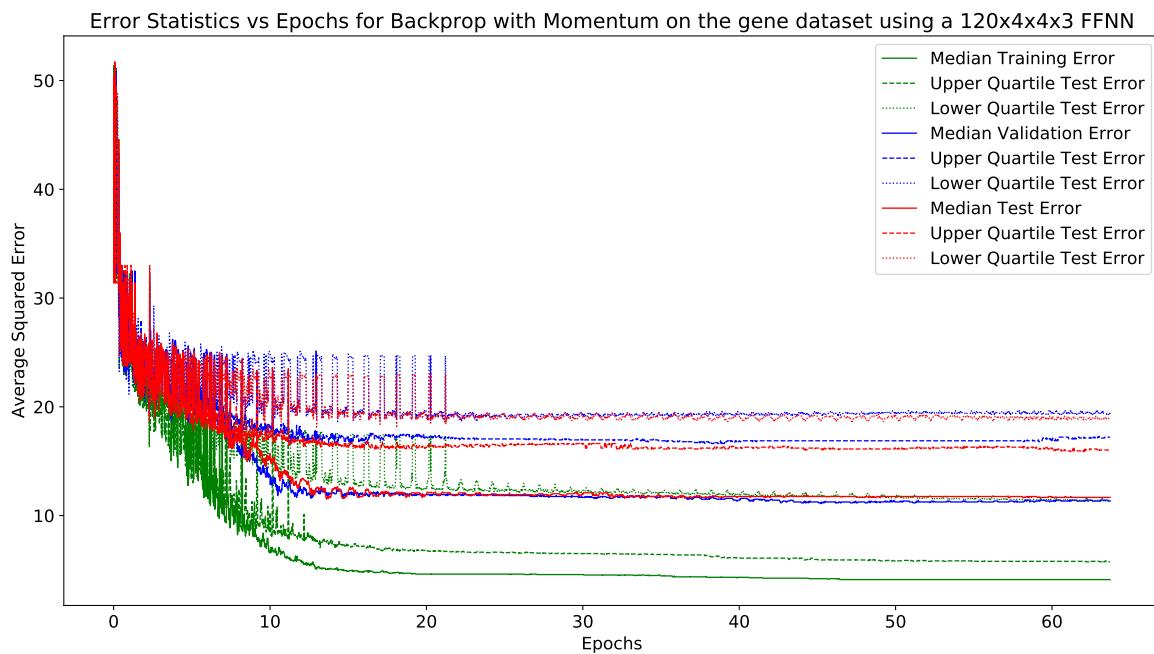
This section displays the results obtained using the gene algorithm.

##### 120 × 4 × 4 × 3 Architecture:

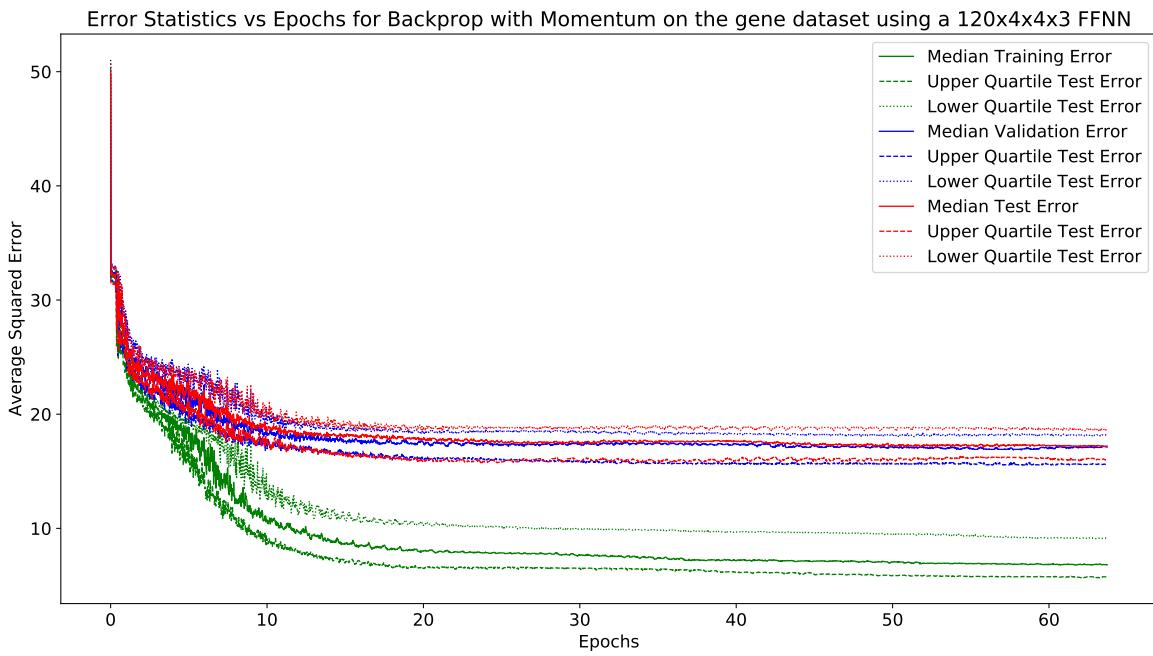
Fig. 269 shows the average and standard deviation of the test, training and validation errors. Fig. 270 shows the median and quartiles of the test, training and validation errors based on the best epoch. Finally, Fig. 271 shows the median and quartiles of the test, training and validation errors per mini-batch.



**Figure 269.** Graph of mean and standard deviation of errors vs epochs



**Figure 270.** Graph of test error vs epochs for the gradient based algorithms



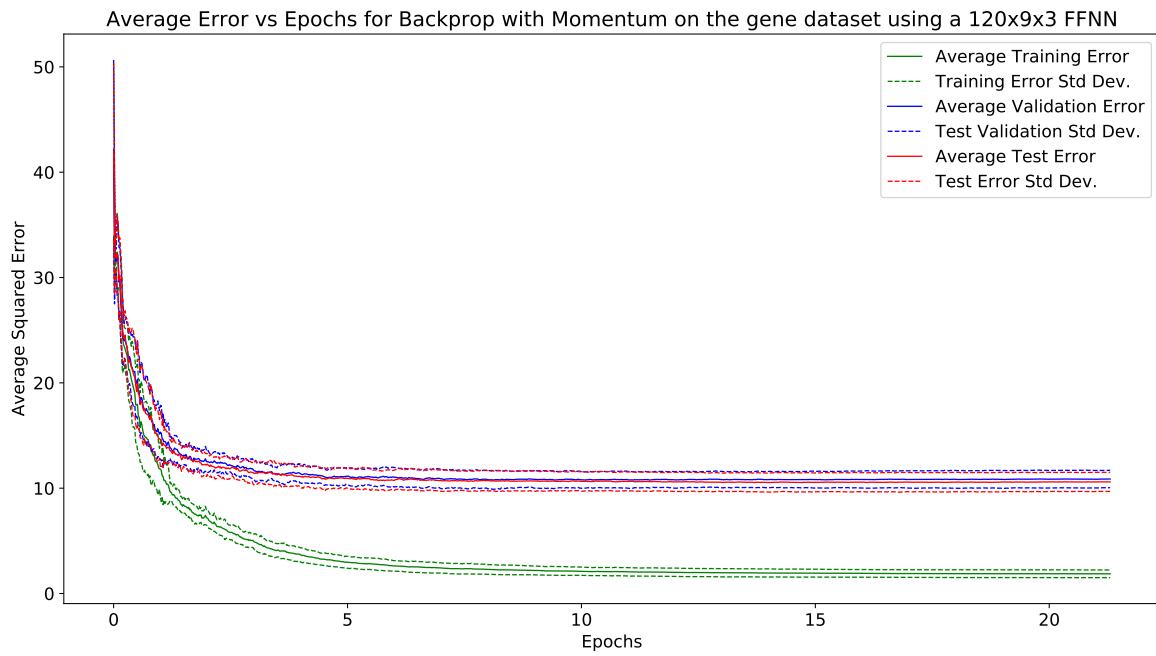
**Figure 271.** Graph of test error vs epochs for the gradient based algorithms

#### 16.4.8 gene

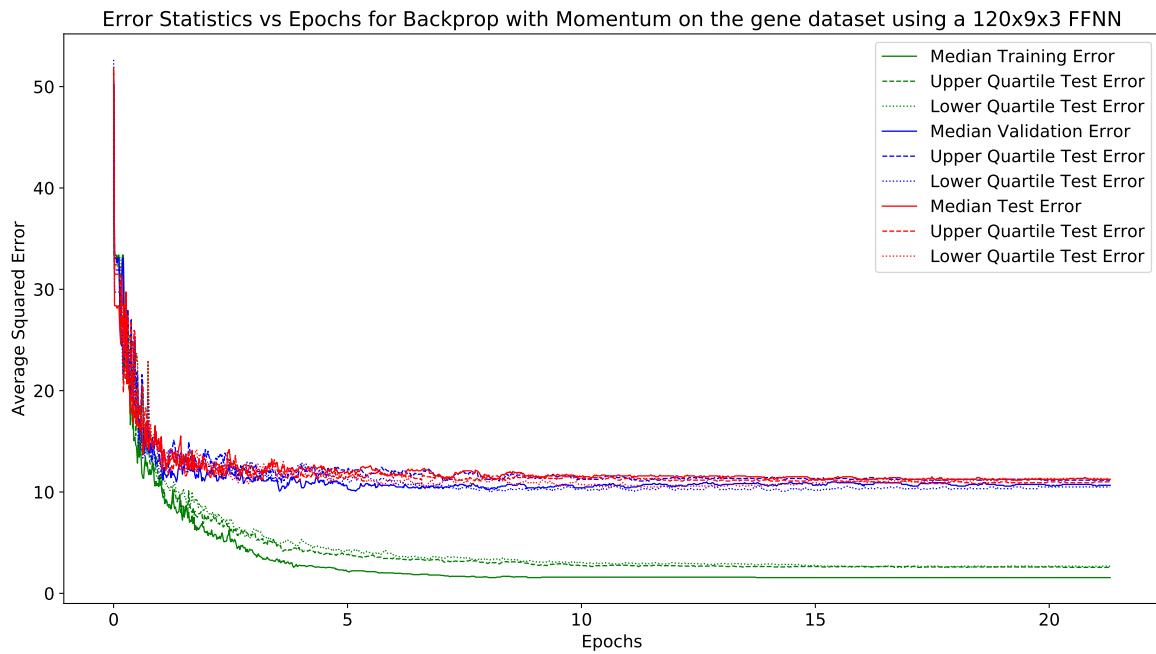
This section displays the results obtained using the gene algorithm.

##### 120 × 9 × 3 Architecture:

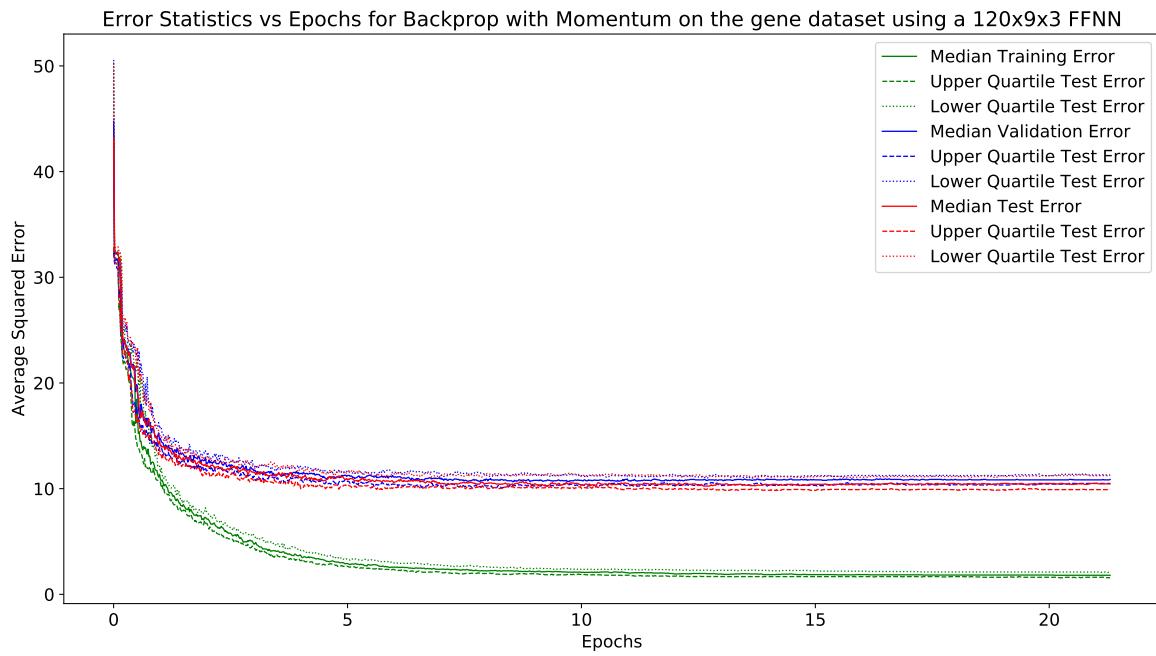
Fig. 272 shows the average and standard deviation of the test, training and validation errors. Fig. 273 shows the median and quartiles of the test, training and validation errors based on the best epoch. Finally, Fig. 274 shows the median and quartiles of the test, training and validation errors per mini-batch.



**Figure 272.** Graph of mean and standard deviation of errors vs epochs



**Figure 273.** Graph of test error vs epochs for the gradient based algorithms



**Figure 274.** Graph of test error vs epochs for the gradient based algorithms

#### 16.4.9 back-propogation

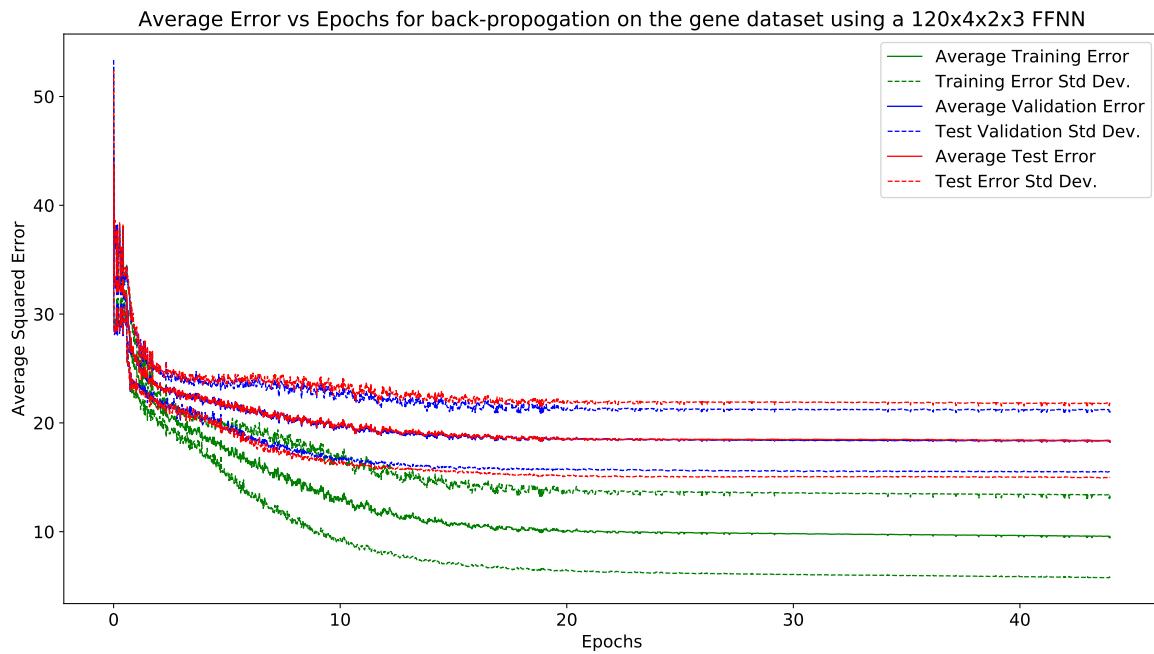
This section displays the results obtained using the back-propogation algorithm.

#### 16.4.10 gene

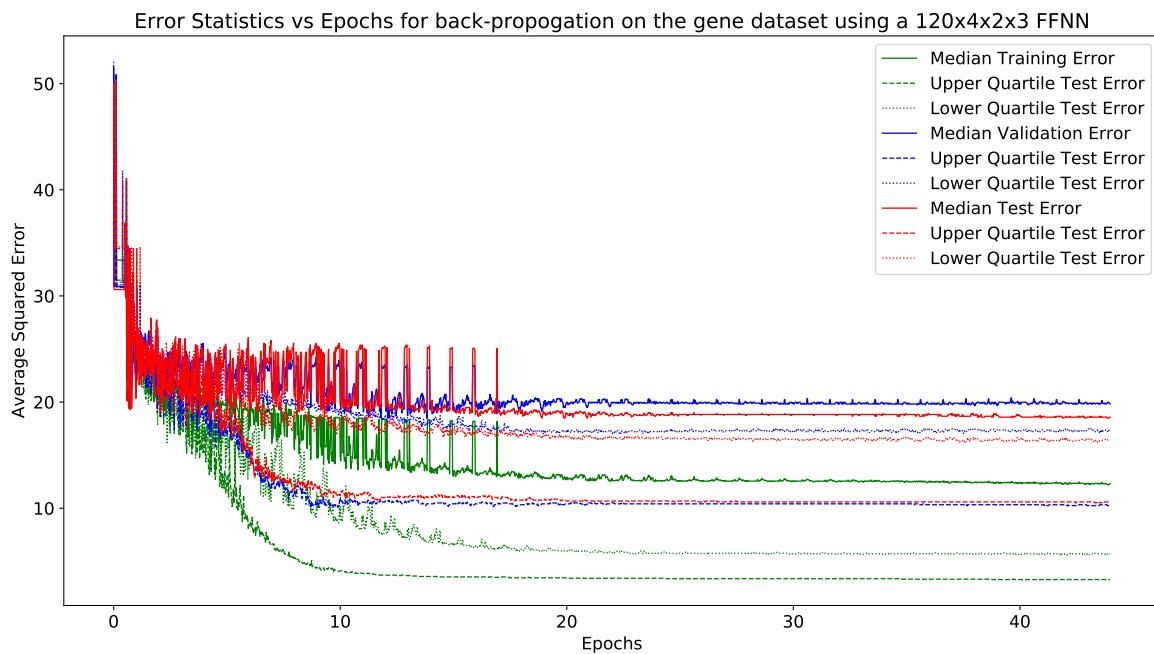
This section displays the results obtained using the gene algorithm.

##### 120 × 4 × 2 × 3 Architecture:

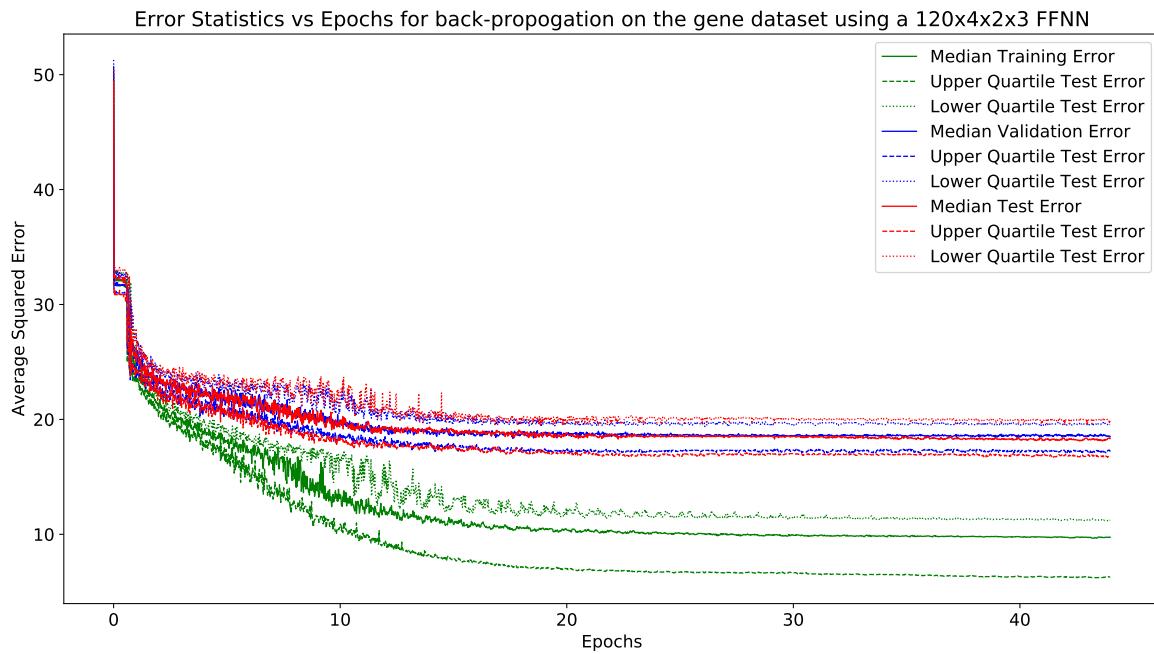
Fig. 275 shows the average and standard deviation of the test, training and validation errors. Fig. 276 shows the median and quartiles of the test, training and validation errors based on the best epoch. Finally, Fig. 277 shows the median and quartiles of the test, training and validation errors per mini-batch.



**Figure 275.** Graph of mean and standard deviation of errors vs epochs



**Figure 276.** Graph of test error vs epochs for the gradient based algorithms



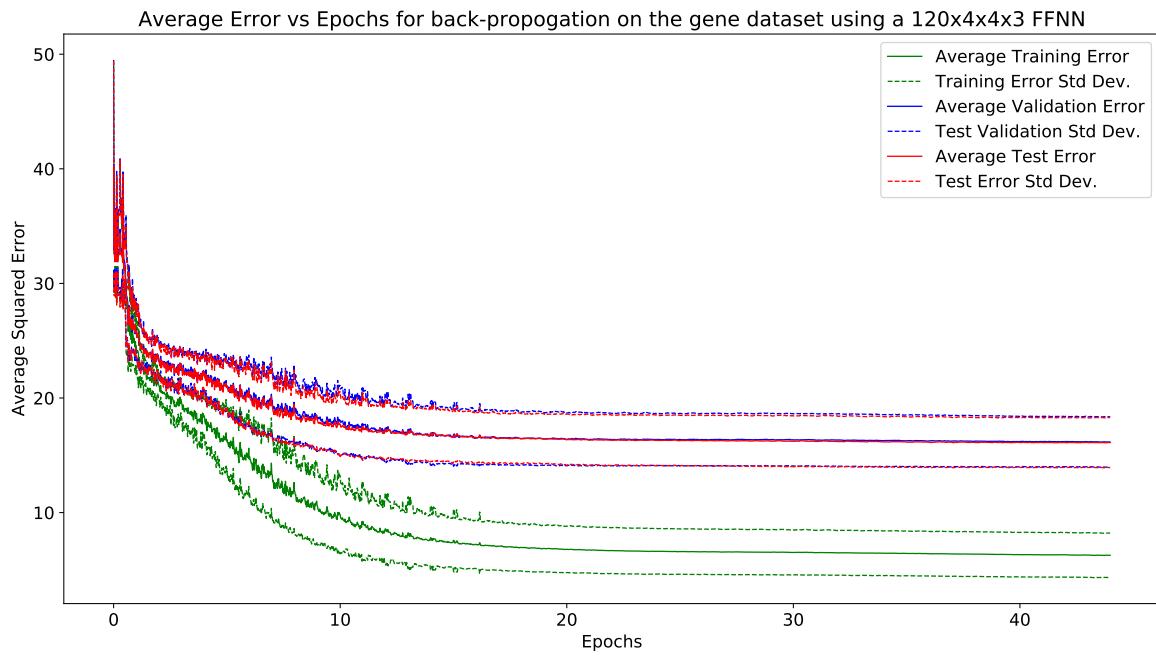
**Figure 277.** Graph of test error vs epochs for the gradient based algorithms

### 16.4.11 gene

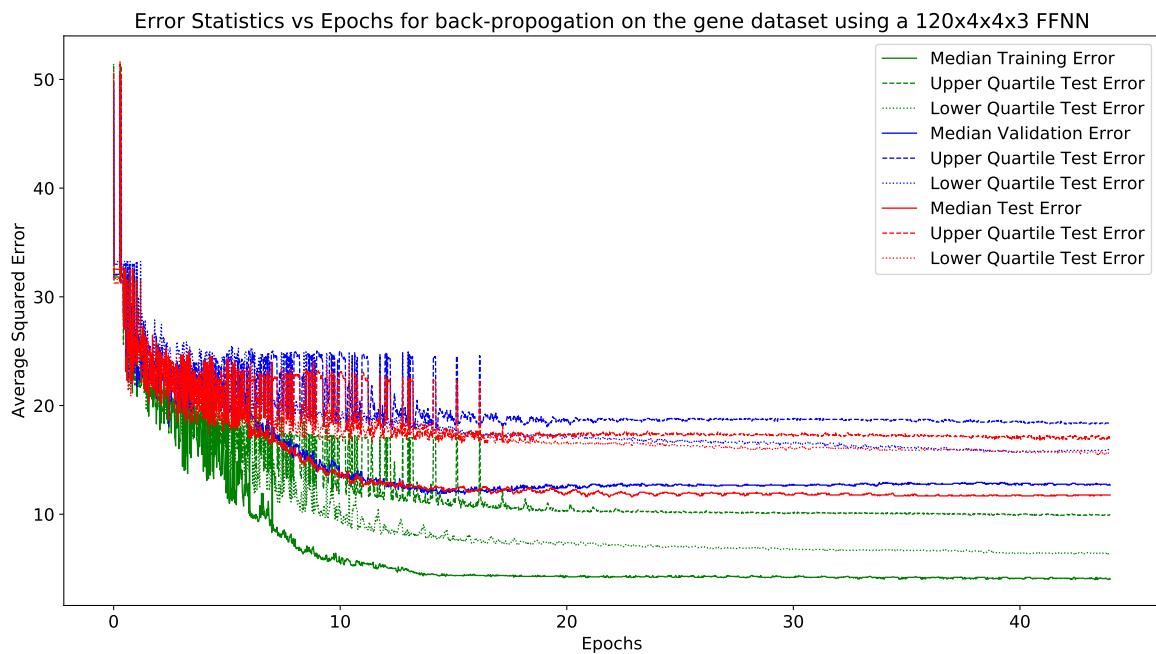
This section displays the results obtained using the gene algorithm.

#### 120 × 4 × 4 × 3 Architecture:

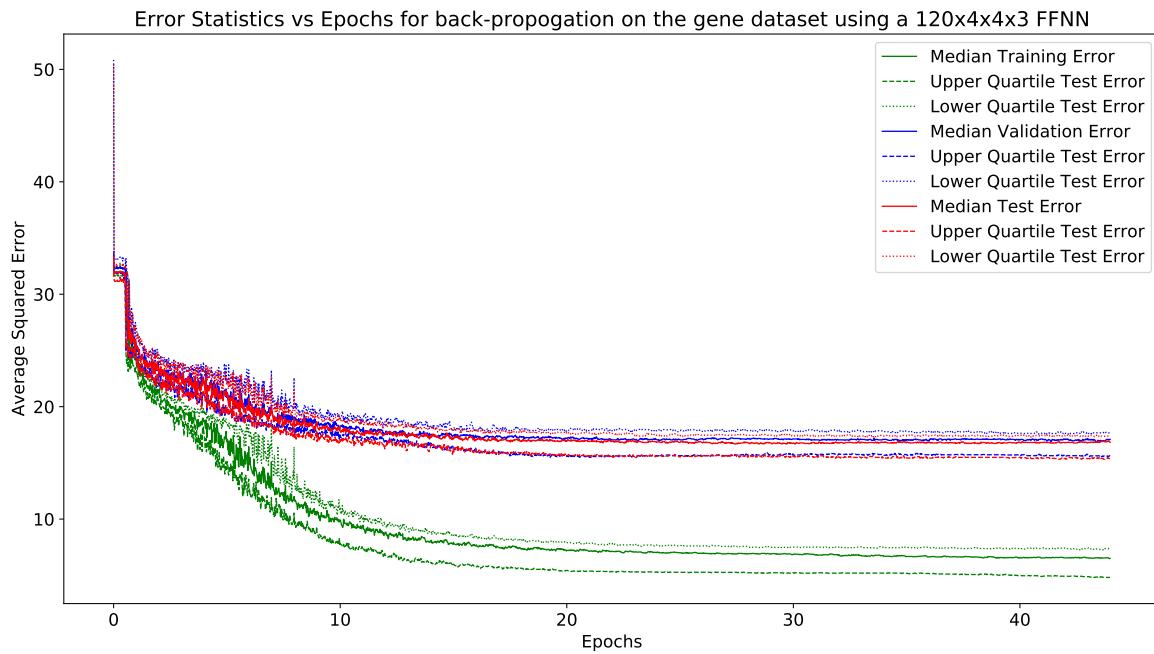
Fig. 278 shows the average and standard deviation of the test, training and validation errors. Fig. 279 shows the median and quartiles of the test, training and validation errors based on the best epoch. Finally, Fig. 280 shows the median and quartiles of the test, training and validation errors per mini-batch.



**Figure 278.** Graph of mean and standard deviation of errors vs epochs



**Figure 279.** Graph of test error vs epochs for the gradient based algorithms



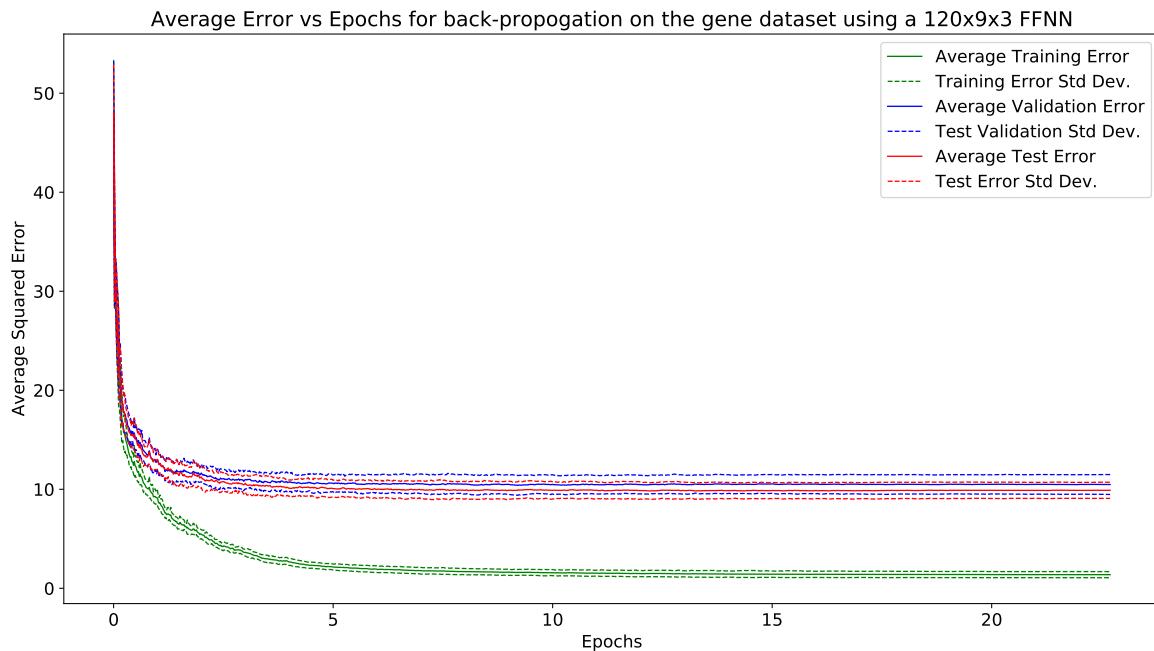
**Figure 280.** Graph of test error vs epochs for the gradient based algorithms

### 16.4.12 gene

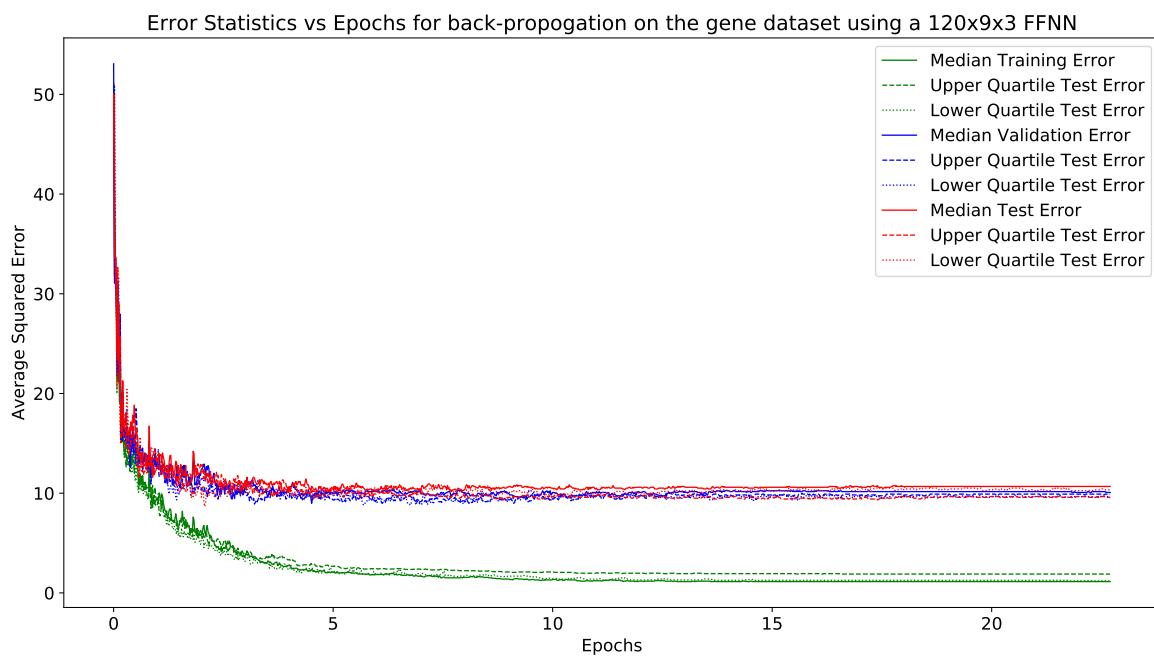
This section displays the results obtained using the gene algorithm.

#### 120 × 9 × 3 Architecture:

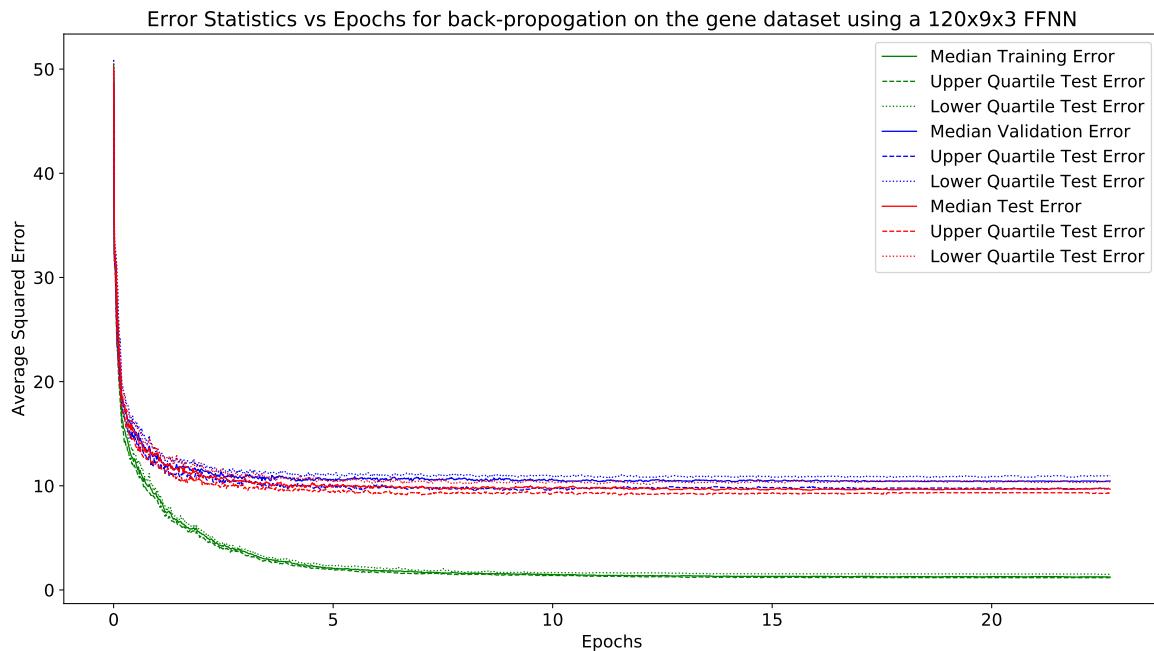
Fig. 281 shows the average and standard deviation of the test, training and validation errors. Fig. 282 shows the median and quartiles of the test, training and validation errors based on the best epoch. Finally, Fig. 283 shows the median and quartiles of the test, training and validation errors per mini-batch.



**Figure 281.** Graph of mean and standard deviation of errors vs epochs



**Figure 282.** Graph of test error vs epochs for the gradient based algorithms



**Figure 283.** Graph of test error vs epochs for the gradient based algorithms

#### 16.4.13 GA1

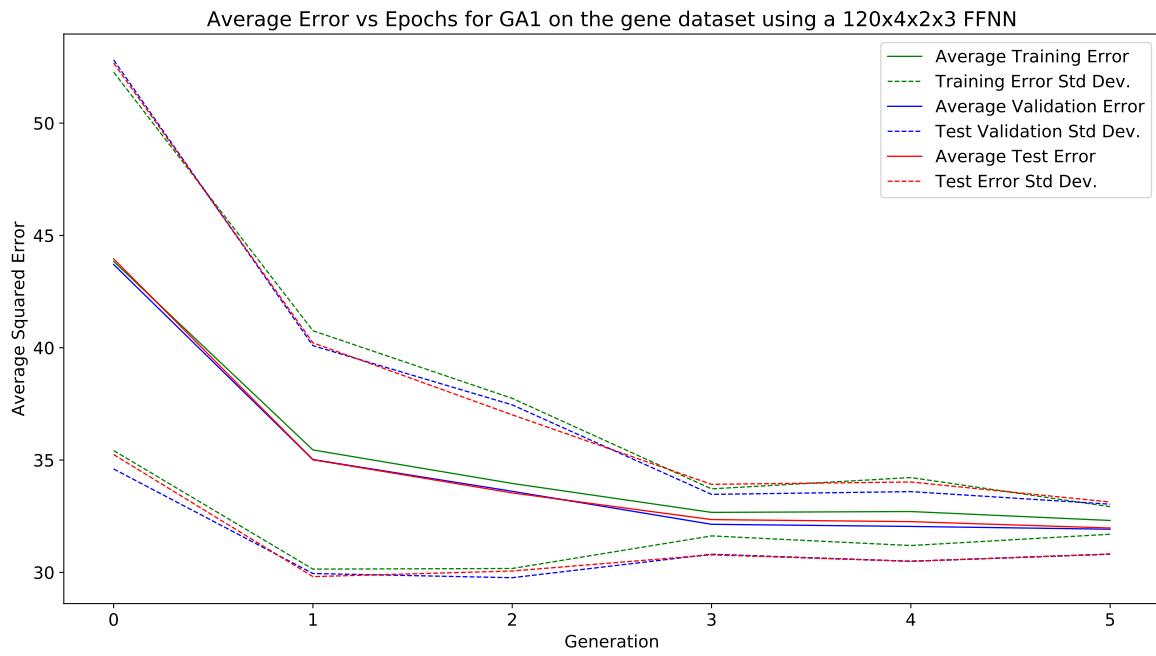
This section displays the results obtained using the GA1 algorithm.

#### 16.4.14 gene

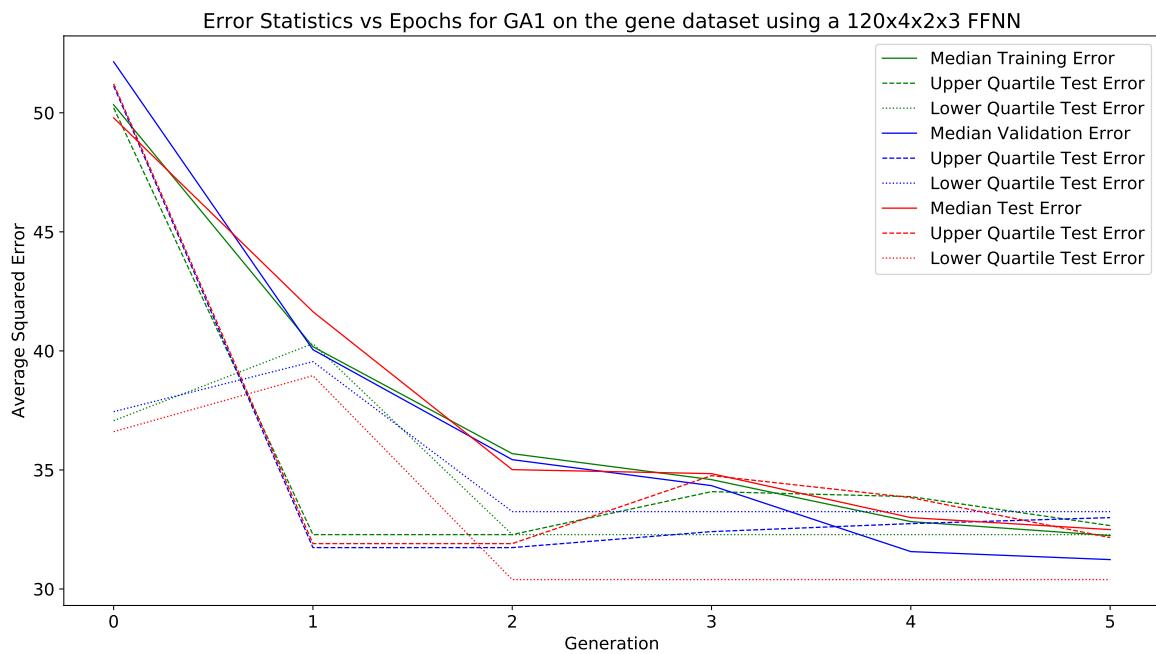
This section displays the results obtained using the gene algorithm.

##### 120 × 4 × 2 × 3 Architecture:

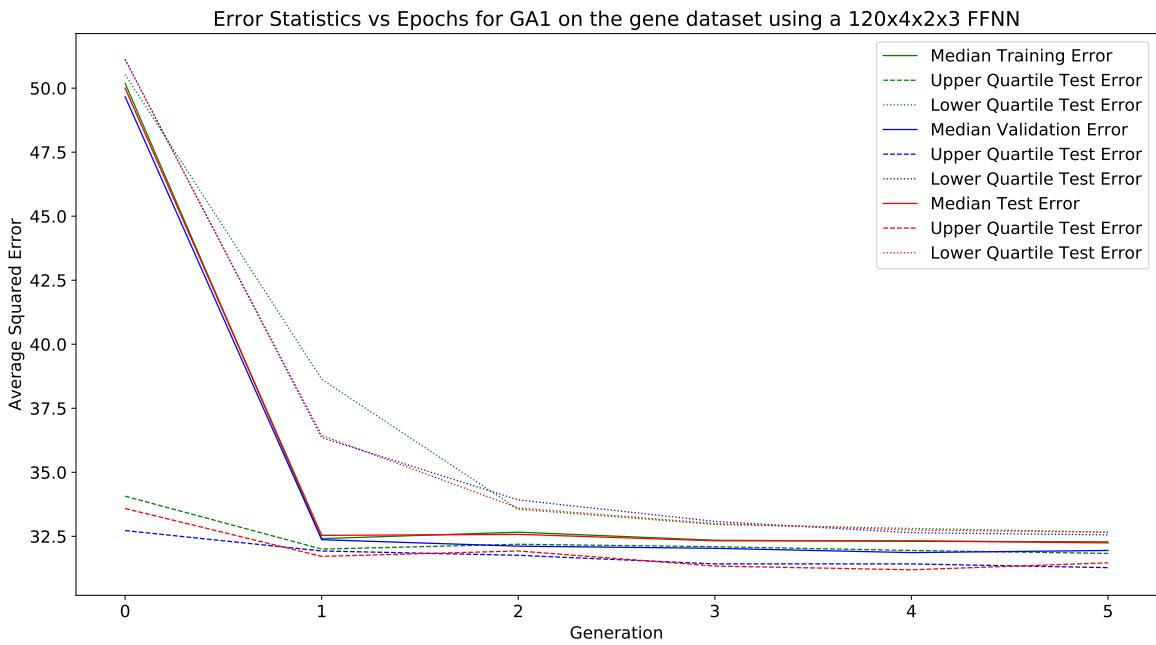
Fig. 284 shows the average and standard deviation of the test, training and validation errors. Fig. 285 shows the median and quartiles of the test, training and validation errors based on the best epoch. Finally, Fig. 286 shows the median and quartiles of the test, training and validation errors per mini-batch.



**Figure 284.** Graph of mean and standard deviation of errors vs epochs



**Figure 285.** Graph of test error vs epochs for the gradient based algorithms



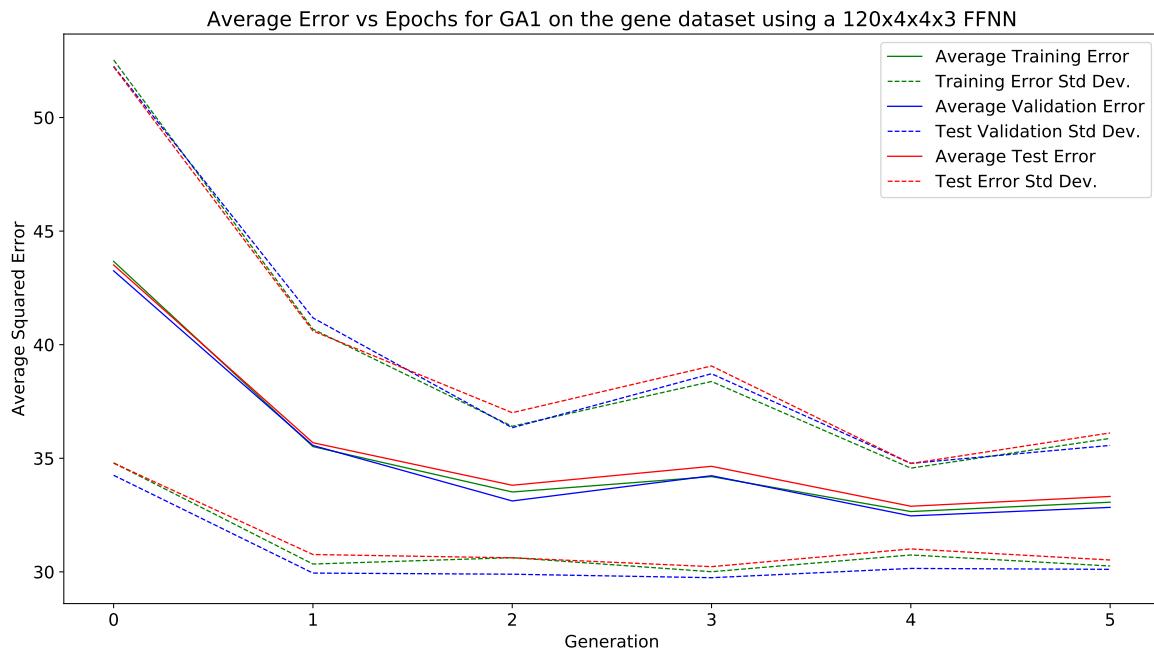
**Figure 286.** Graph of test error vs epochs for the gradient based algorithms

### 16.4.15 gene

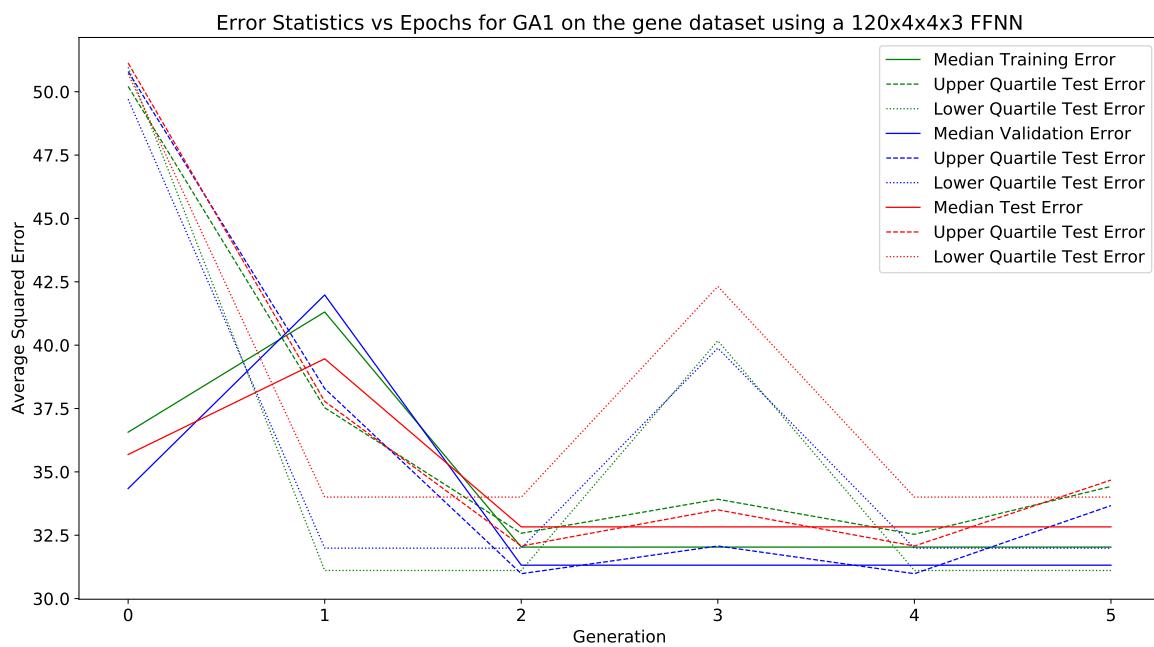
This section displays the results obtained using the gene algorithm.

#### 120 × 4 × 4 × 3 Architecture:

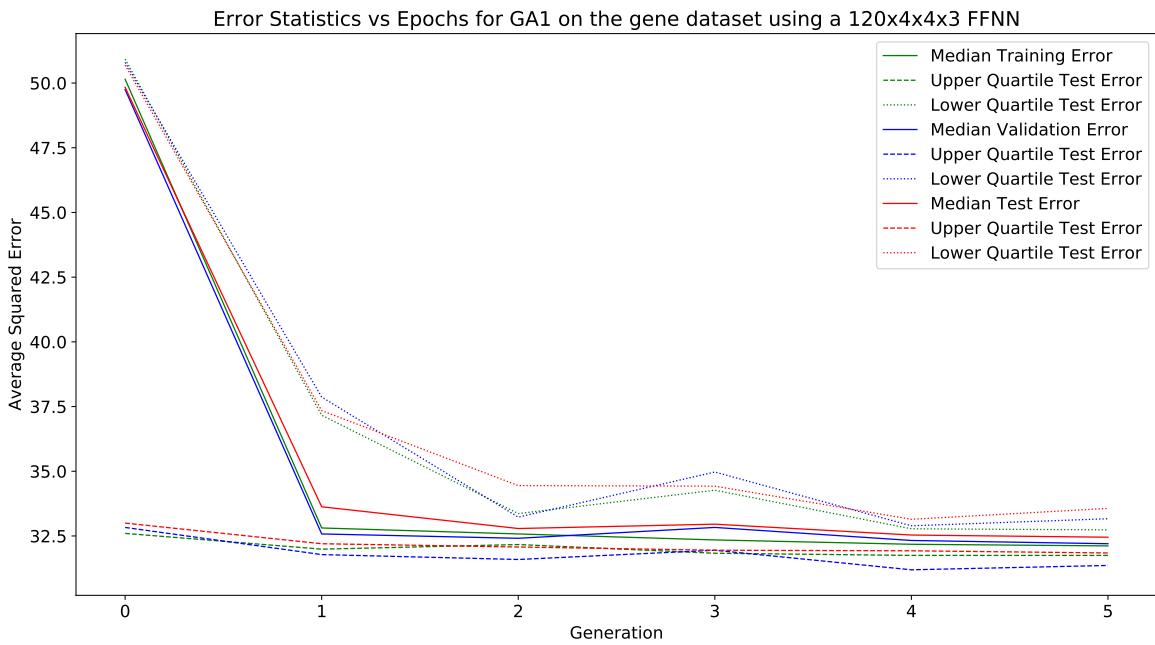
Fig. 287 shows the average and standard deviation of the test, training and validation errors. Fig. 288 shows the median and quartiles of the test, training and validation errors based on the best epoch. Finally, Fig. 289 shows the median and quartiles of the test, training and validation errors per mini-batch.



**Figure 287.** Graph of mean and standard deviation of errors vs epochs



**Figure 288.** Graph of test error vs epochs for the gradient based algorithms



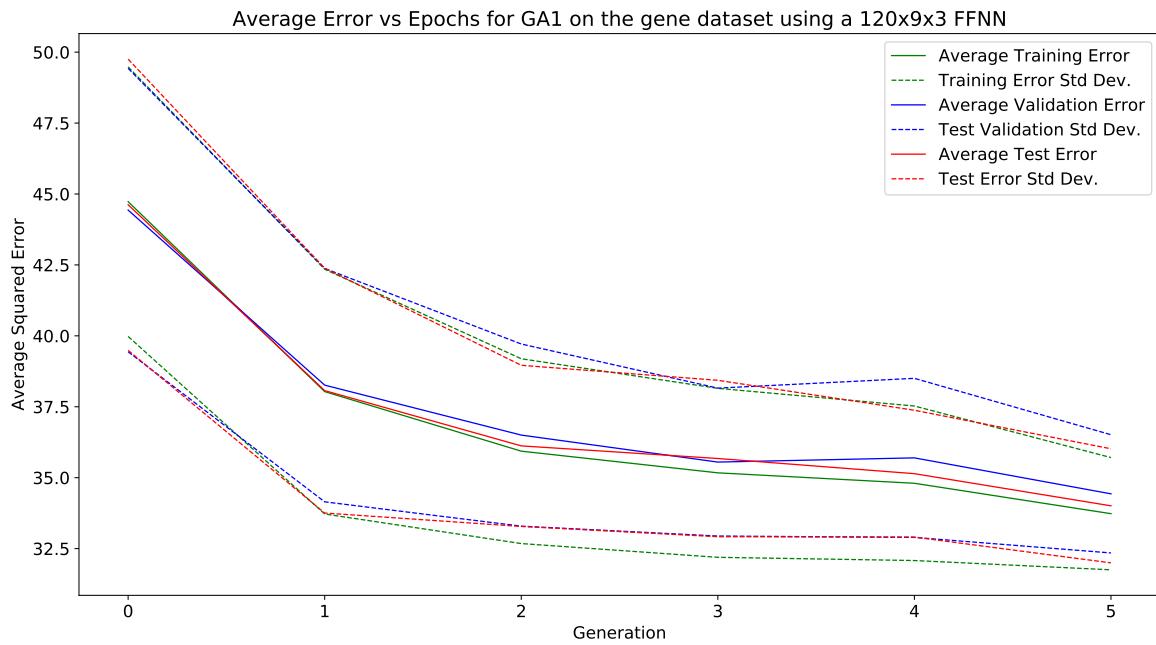
**Figure 289.** Graph of test error vs epochs for the gradient based algorithms

### 16.4.16 gene

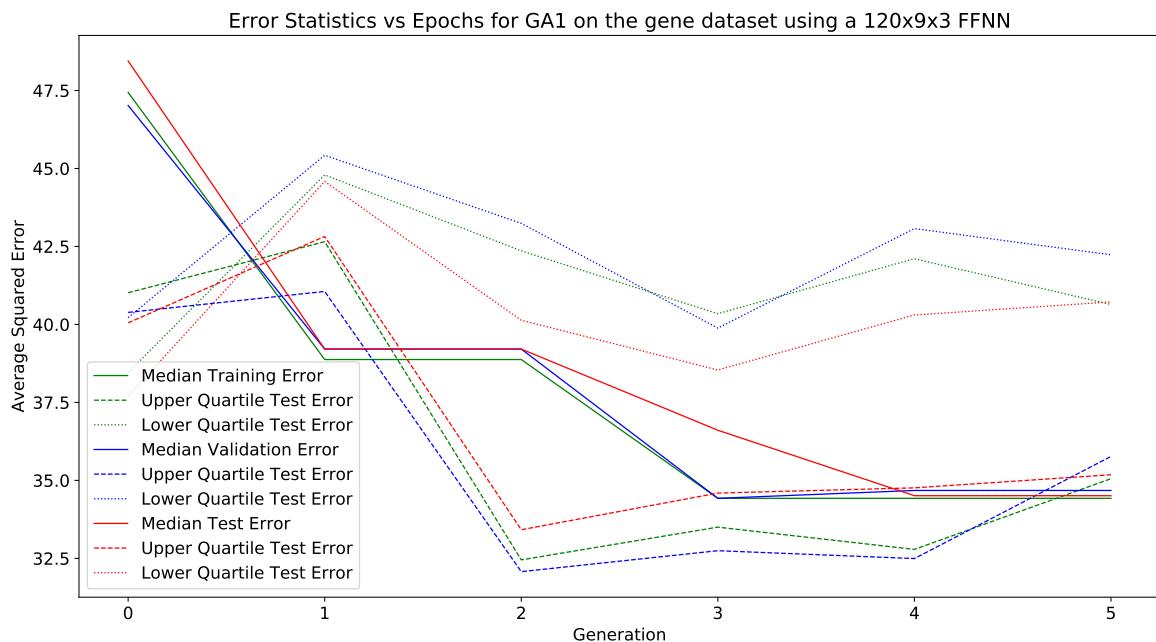
This section displays the results obtained using the gene algorithm.

#### 120 × 9 × 3 Architecture:

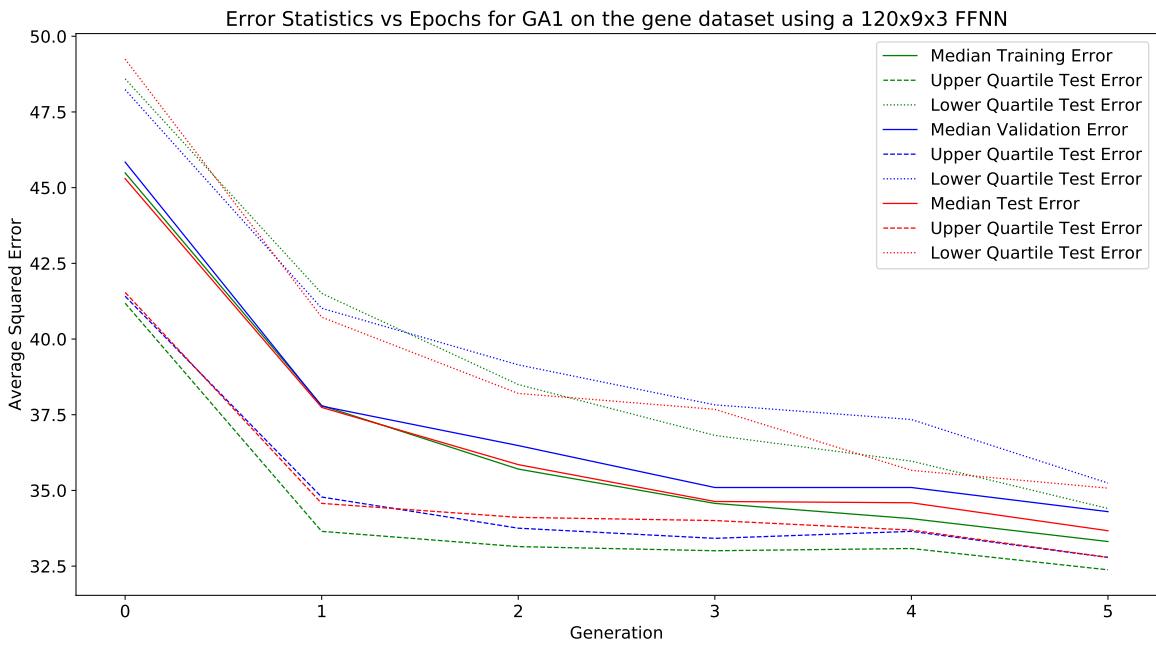
Fig. 290 shows the average and standard deviation of the test, training and validation errors. Fig. 291 shows the median and quartiles of the test, training and validation errors based on the best epoch. Finally, Fig. 292 shows the median and quartiles of the test, training and validation errors per mini-batch.



**Figure 290.** Graph of mean and standard deviation of errors vs epochs



**Figure 291.** Graph of test error vs epochs for the gradient based algorithms



**Figure 292.** Graph of test error vs epochs for the gradient based algorithms

#### 16.4.17 GA2

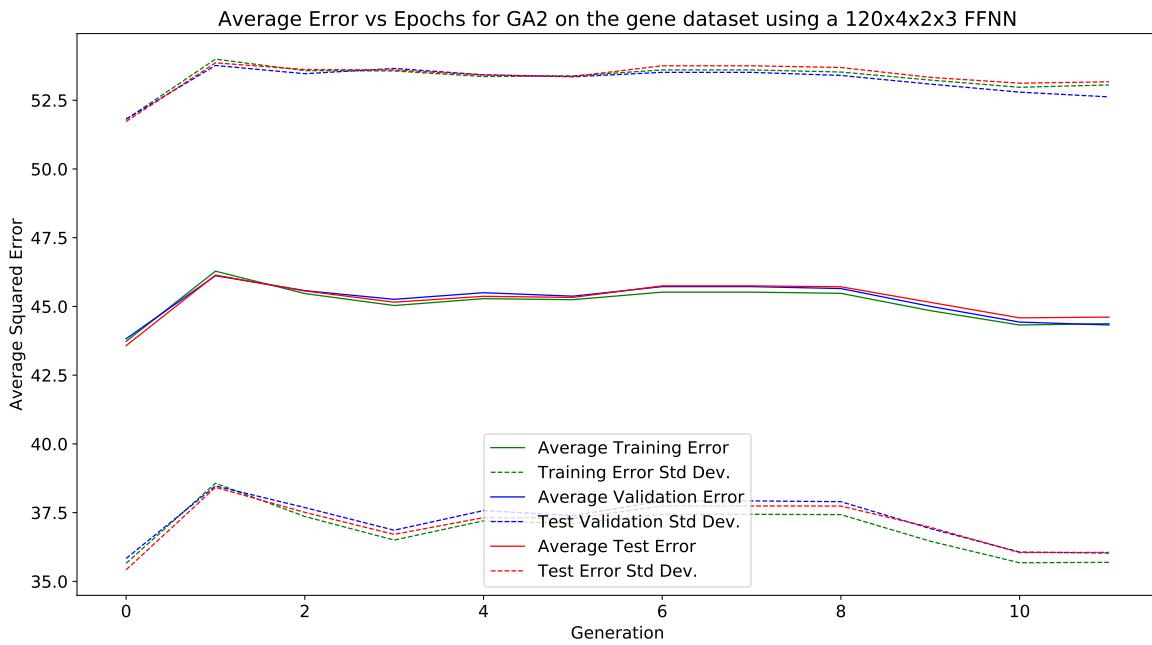
This section displays the results obtained using the GA2 algorithm.

#### 16.4.18 gene

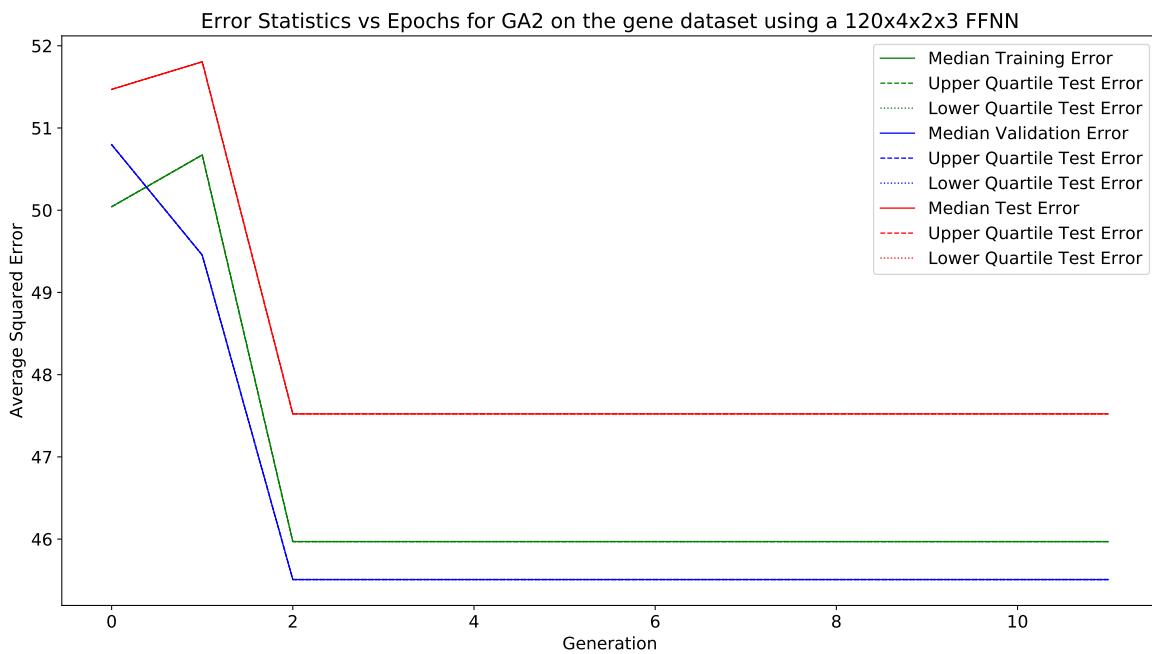
This section displays the results obtained using the gene algorithm.

#### 120 × 4 × 2 × 3 Architecture:

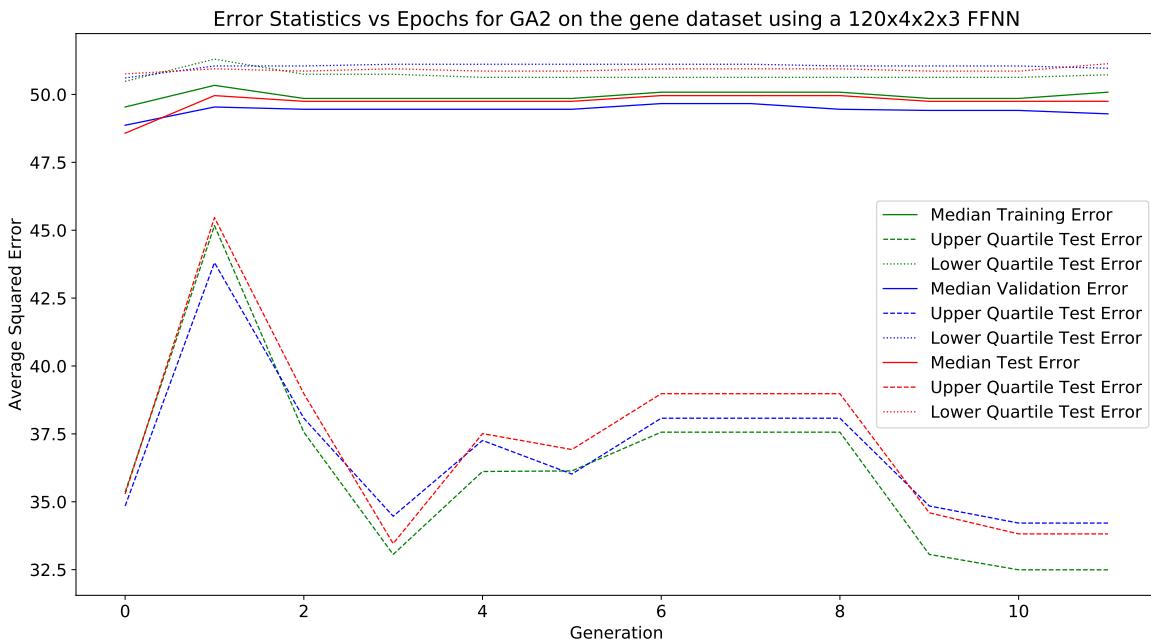
Fig. 293 shows the average and standard deviation of the test, training and validation errors. Fig. 294 shows the median and quartiles of the test, training and validation errors based on the best epoch. Finally, Fig. 295 shows the median and quartiles of the test, training and validation errors per mini-batch.



**Figure 293.** Graph of mean and standard deviation of errors vs epochs



**Figure 294.** Graph of test error vs epochs for the gradient based algorithms



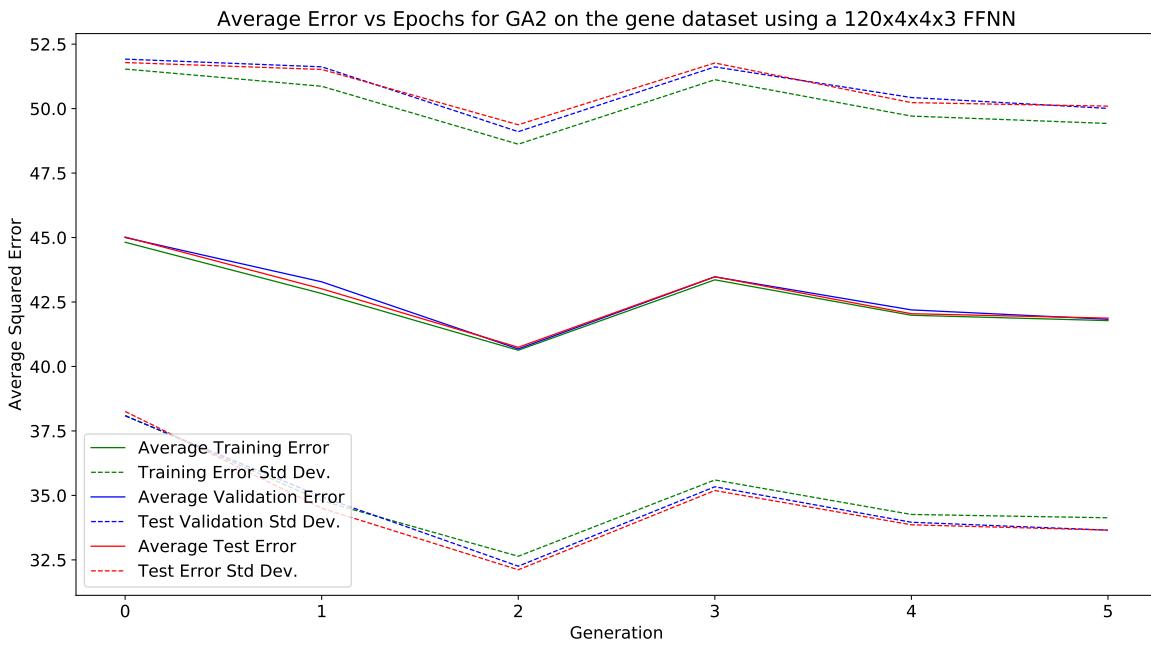
**Figure 295.** Graph of test error vs epochs for the gradient based algorithms

### 16.4.19 gene

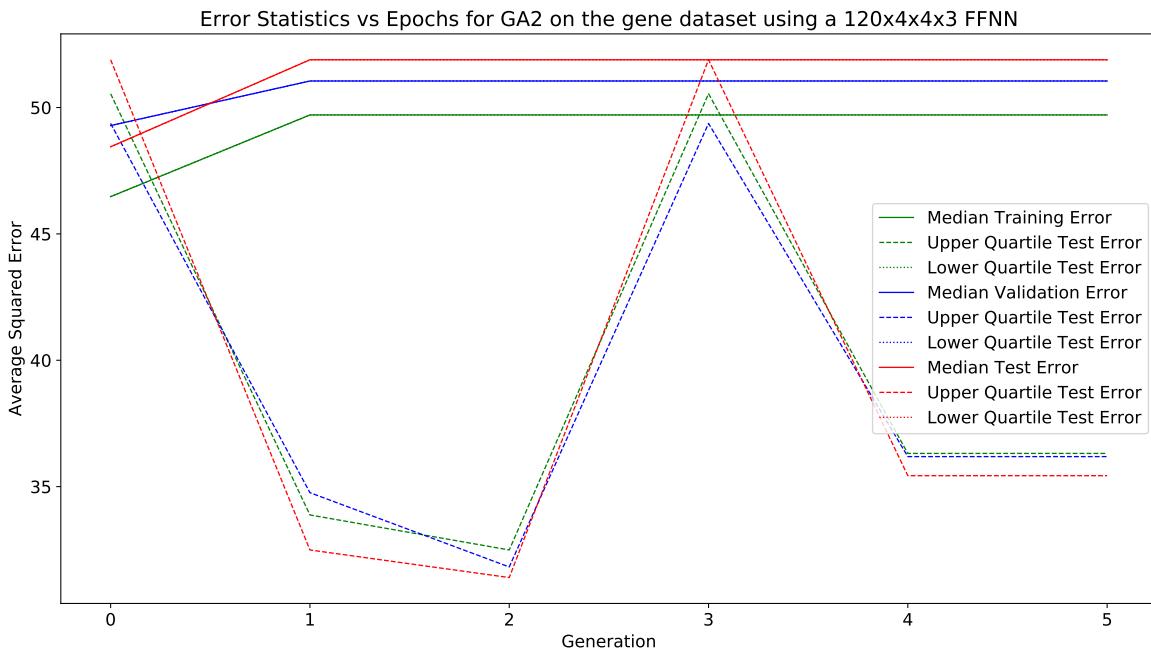
This section displays the results obtained using the gene algorithm.

#### 120 × 4 × 4 × 3 Architecture:

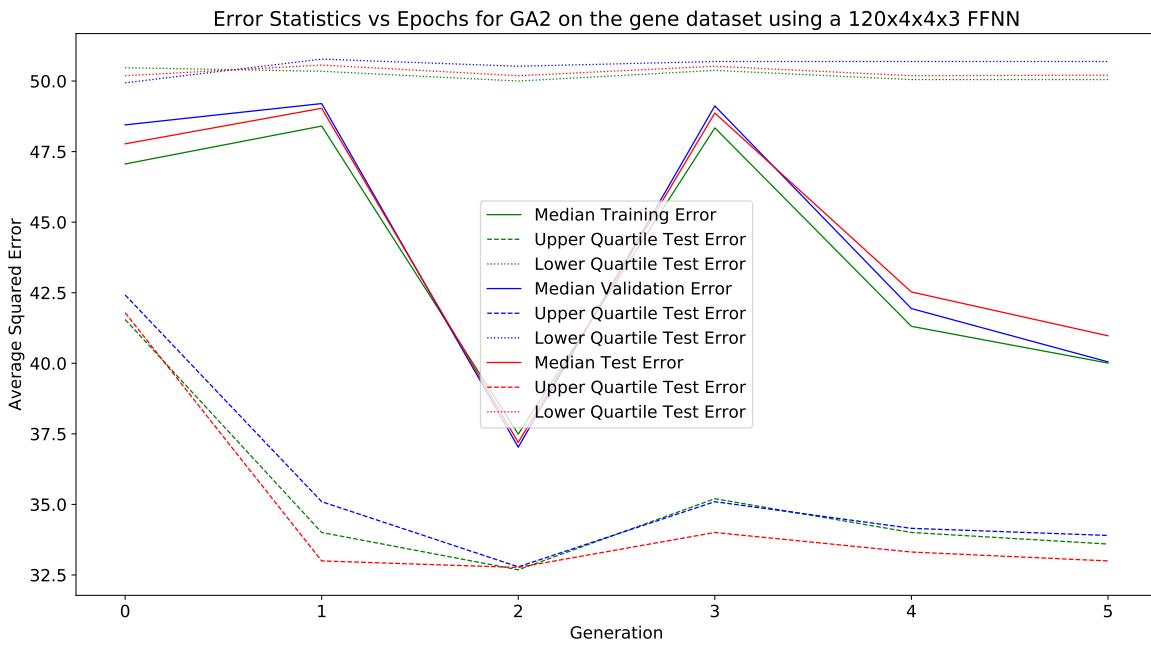
Fig. 296 shows the average and standard deviation of the test, training and validation errors. Fig. 297 shows the median and quartiles of the test, training and validation errors based on the best epoch. Finally, Fig. 298 shows the median and quartiles of the test, training and validation errors per mini-batch.



**Figure 296.** Graph of mean and standard deviation of errors vs epochs



**Figure 297.** Graph of test error vs epochs for the gradient based algorithms



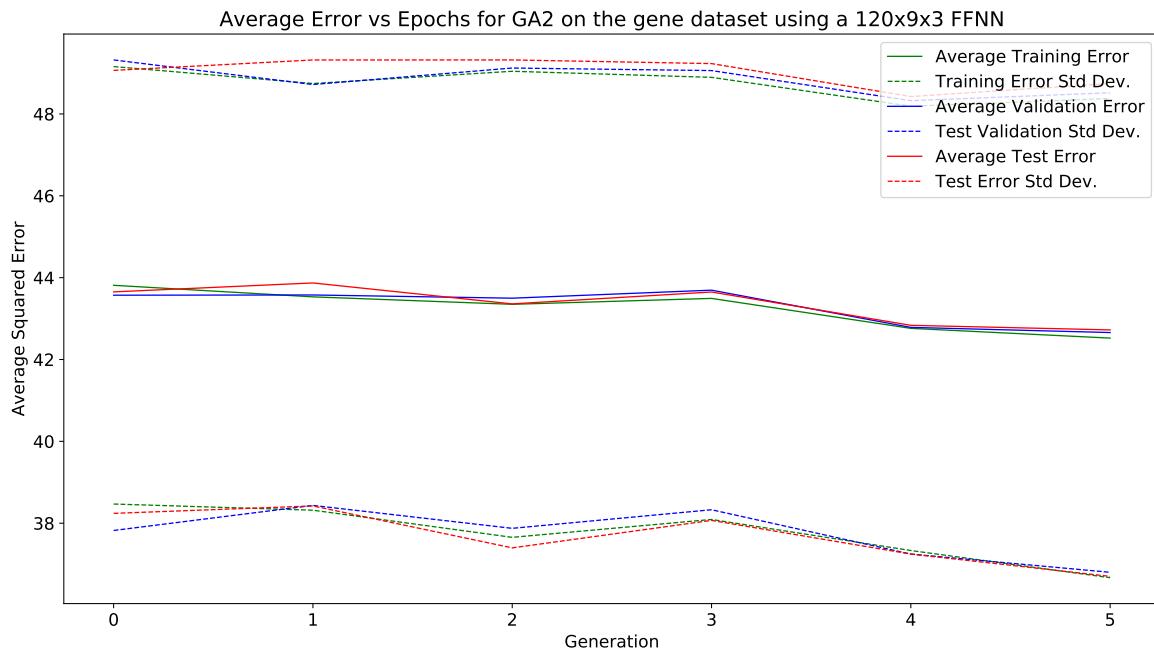
**Figure 298.** Graph of test error vs epochs for the gradient based algorithms

#### 16.4.20 gene

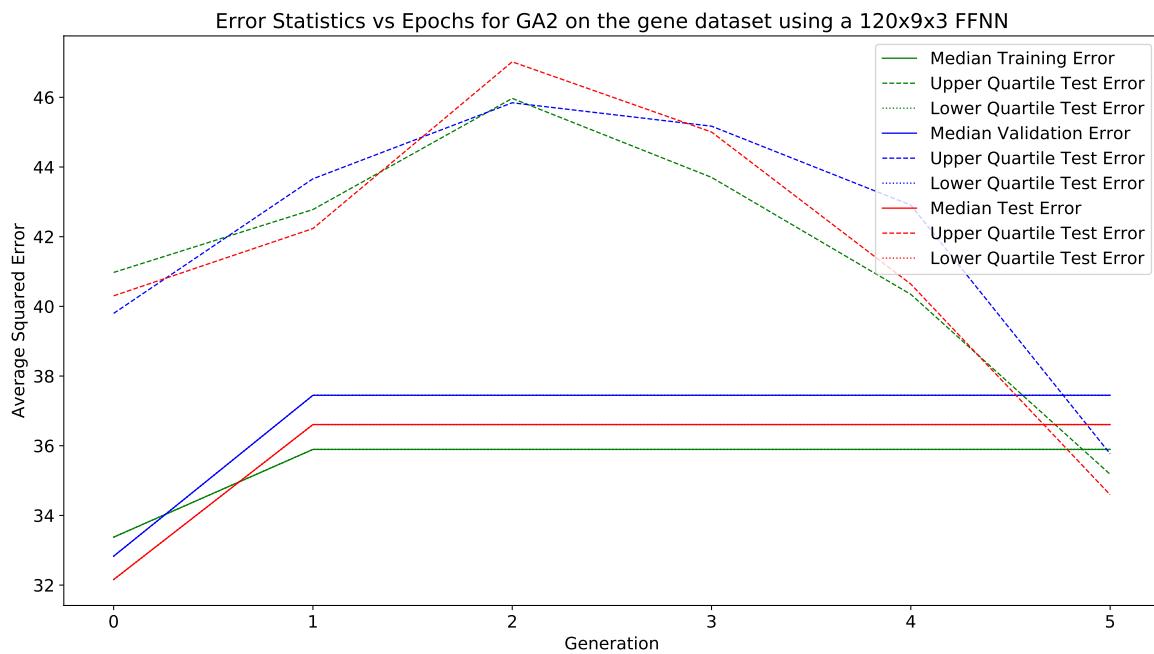
This section displays the results obtained using the gene algorithm.

##### 120 × 9 × 3 Architecture:

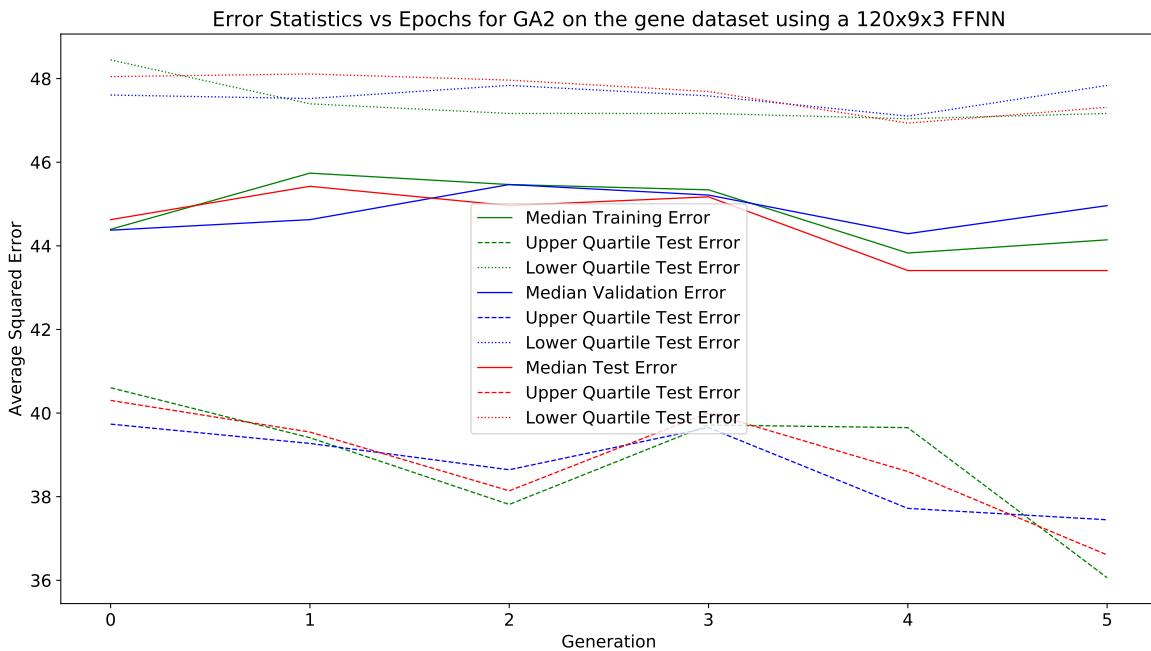
Fig. 299 shows the average and standard deviation of the test, training and validation errors. Fig. 300 shows the median and quartiles of the test, training and validation errors based on the best epoch. Finally, Fig. 301 shows the median and quartiles of the test, training and validation errors per mini-batch.



**Figure 299.** Graph of mean and standard deviation of errors vs epochs



**Figure 300.** Graph of test error vs epochs for the gradient based algorithms



**Figure 301.** Graph of test error vs epochs for the gradient based algorithms

#### 16.4.21 GA3

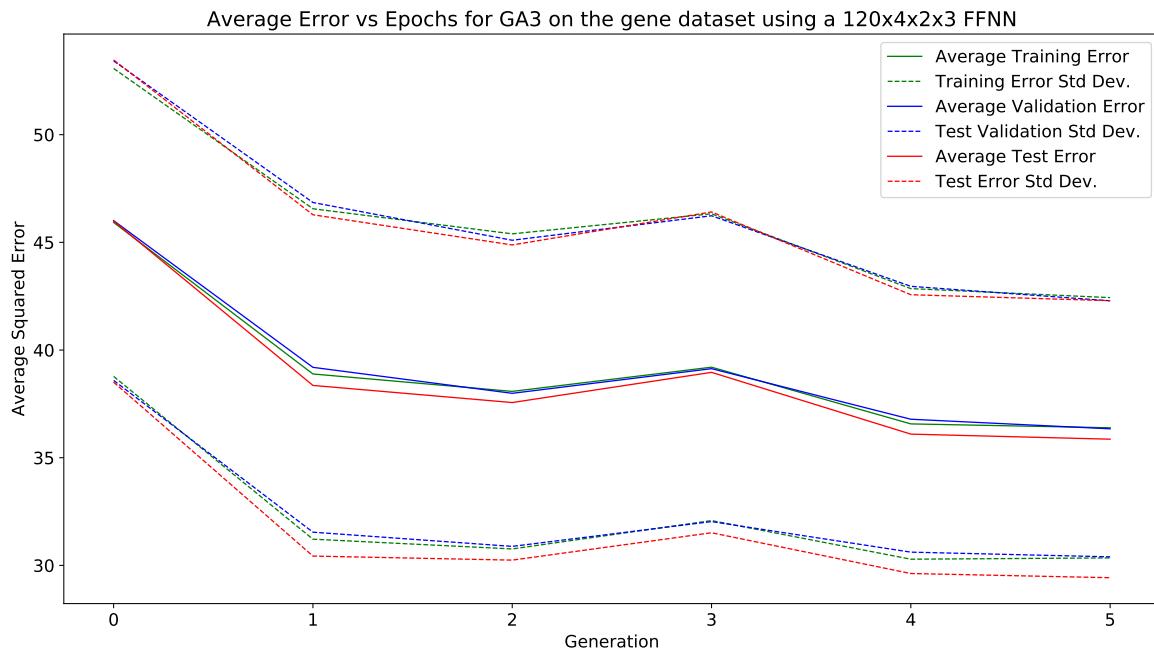
This section displays the results obtained using the GA3 algorithm.

#### 16.4.22 gene

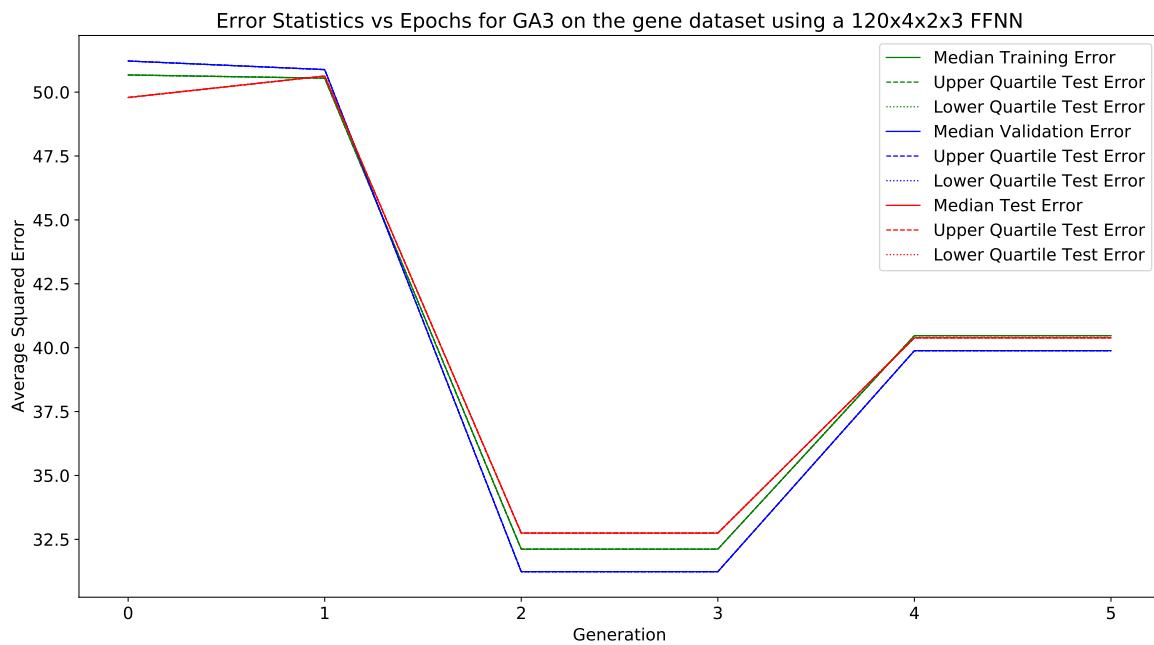
This section displays the results obtained using the gene algorithm.

##### 120 × 4 × 2 × 3 Architecture:

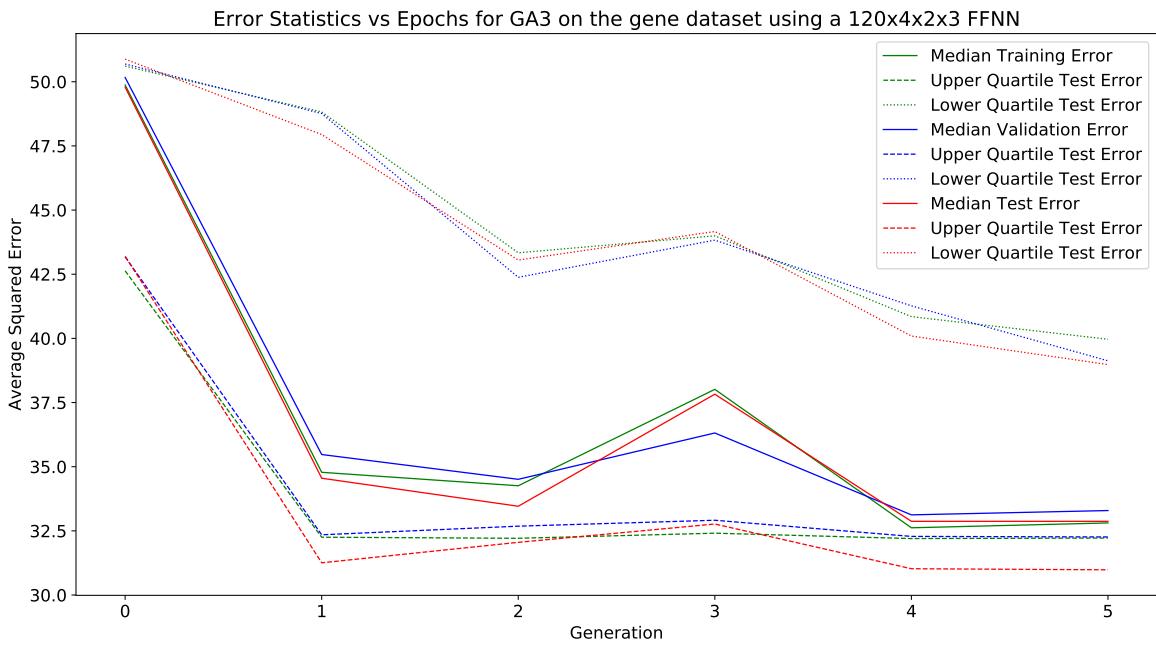
Fig. 302 shows the average and standard deviation of the test, training and validation errors. Fig. 303 shows the median and quartiles of the test, training and validation errors based on the best epoch. Finally, Fig. 304 shows the median and quartiles of the test, training and validation errors per mini-batch.



**Figure 302.** Graph of mean and standard deviation of errors vs epochs



**Figure 303.** Graph of test error vs epochs for the gradient based algorithms



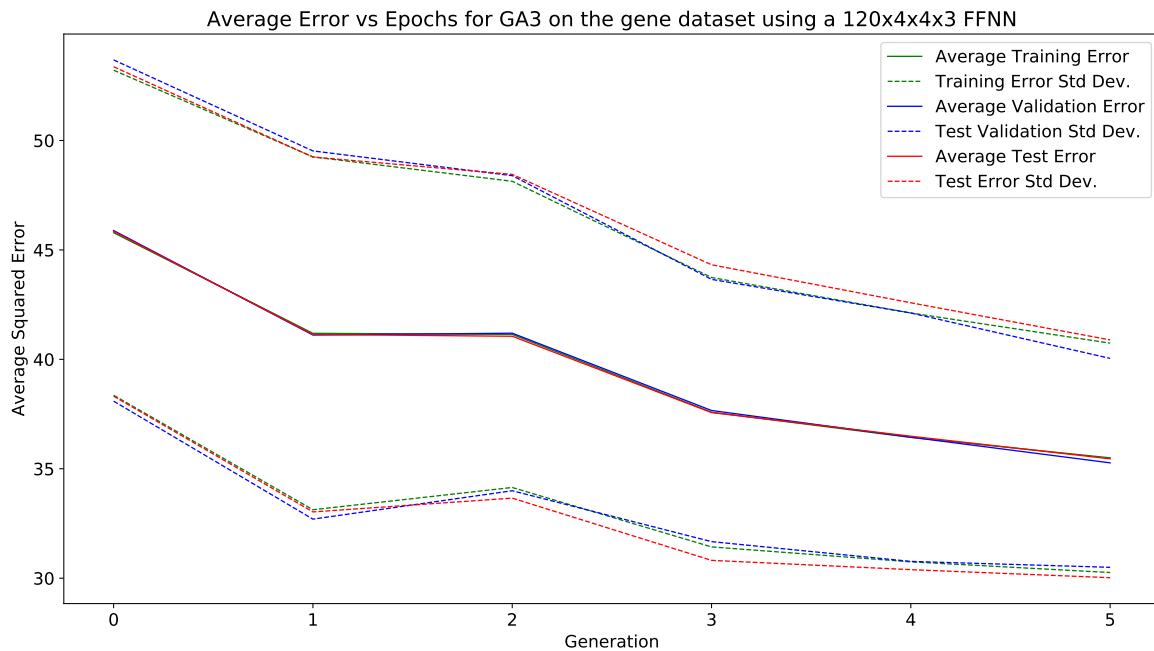
**Figure 304.** Graph of test error vs epochs for the gradient based algorithms

### 16.4.23 gene

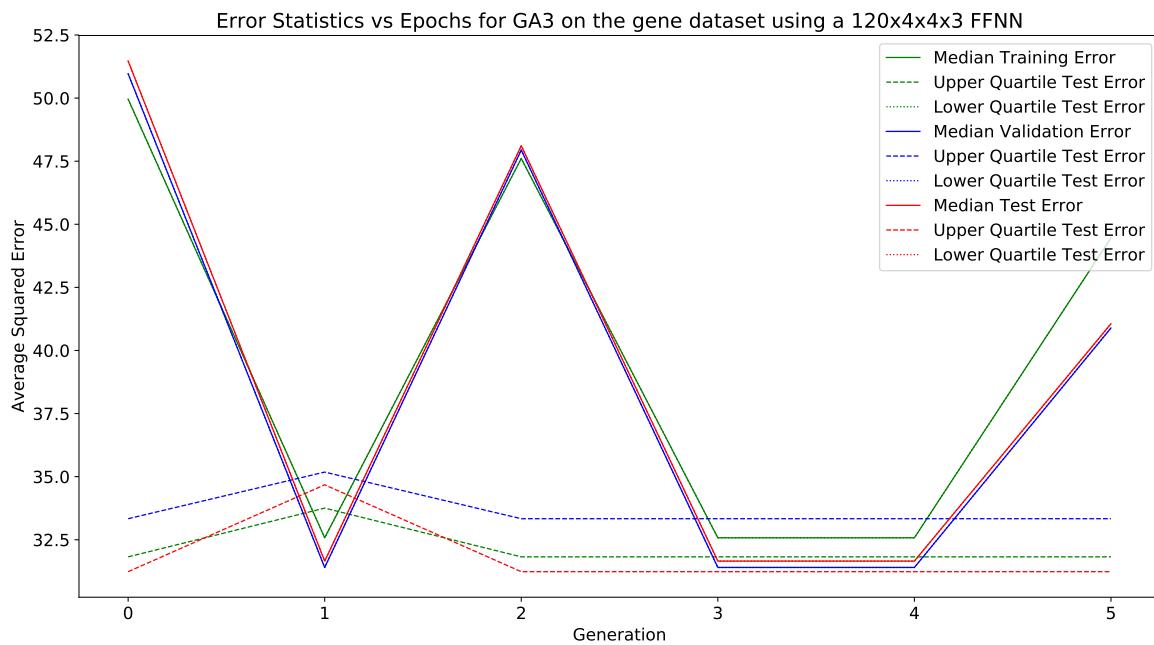
This section displays the results obtained using the gene algorithm.

#### 120 × 4 × 4 × 3 Architecture:

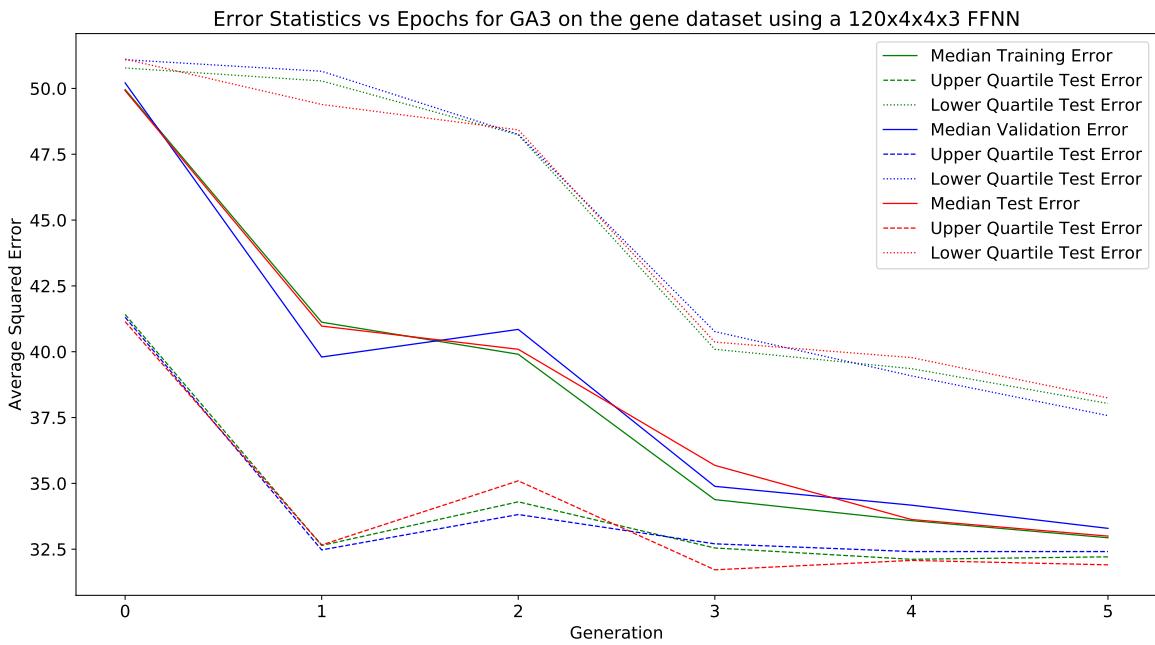
Fig. 305 shows the average and standard deviation of the test, training and validation errors. Fig. 306 shows the median and quartiles of the test, training and validation errors based on the best epoch. Finally, Fig. 307 shows the median and quartiles of the test, training and validation errors per mini-batch.



**Figure 305.** Graph of mean and standard deviation of errors vs epochs



**Figure 306.** Graph of test error vs epochs for the gradient based algorithms



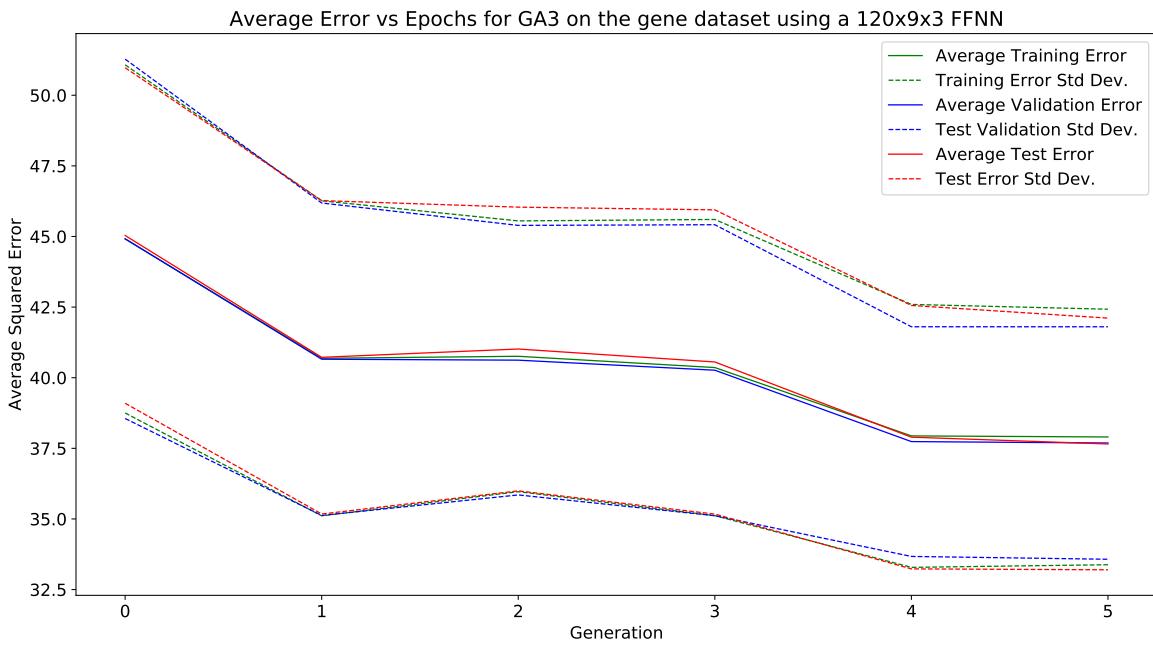
**Figure 307.** Graph of test error vs epochs for the gradient based algorithms

#### 16.4.24 gene

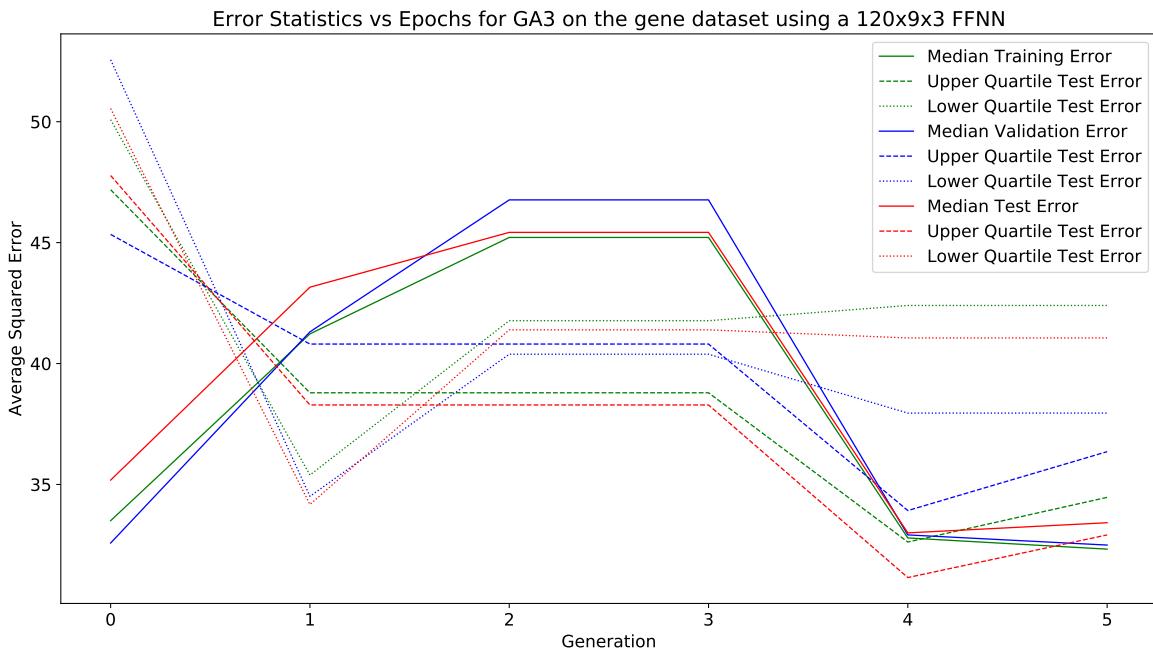
This section displays the results obtained using the gene algorithm.

##### 120 × 9 × 3 Architecture:

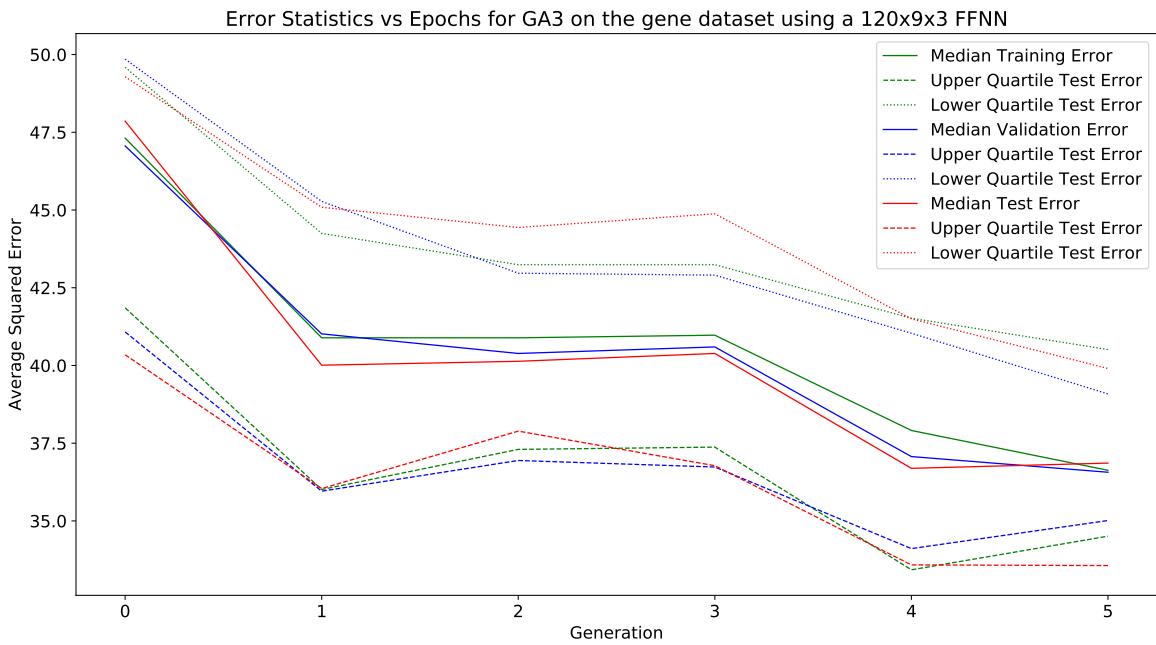
Fig. 308 shows the average and standard deviation of the test, training and validation errors. Fig. 309 shows the median and quartiles of the test, training and validation errors based on the best epoch. Finally, Fig. 310 shows the median and quartiles of the test, training and validation errors per mini-batch.



**Figure 308.** Graph of mean and standard deviation of errors vs epochs



**Figure 309.** Graph of test error vs epochs for the gradient based algorithms



**Figure 310.** Graph of test error vs epochs for the gradient based algorithms

#### 16.4.25 iRPROP-

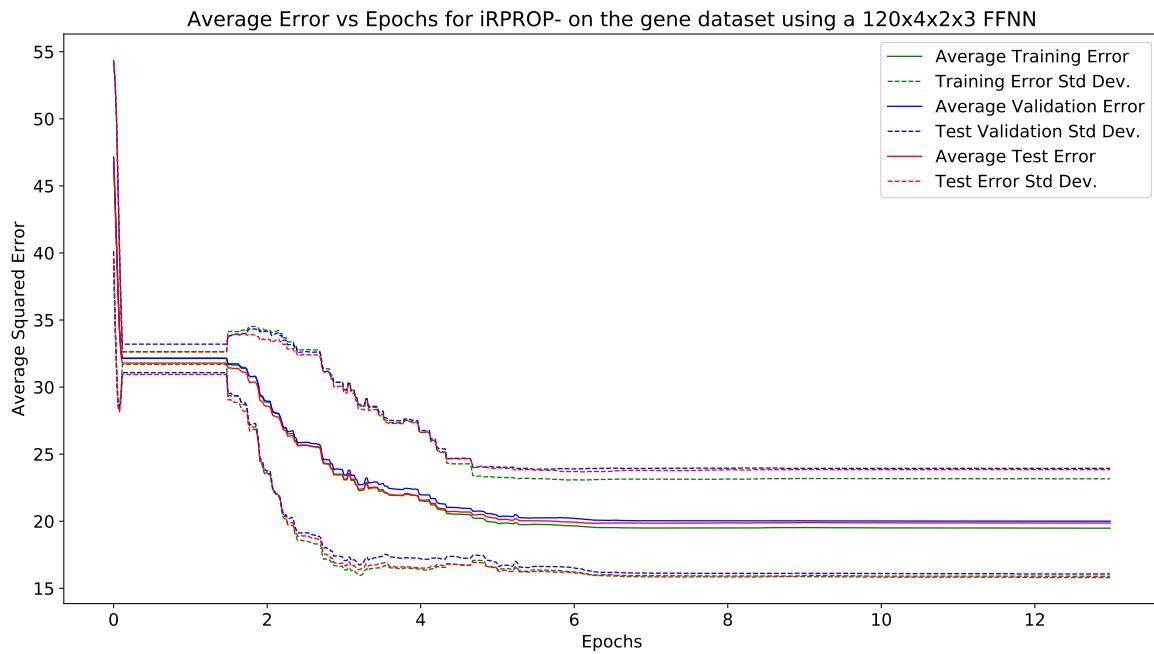
This section displays the results obtained using the iRPROP- algorithm.

#### 16.4.26 gene

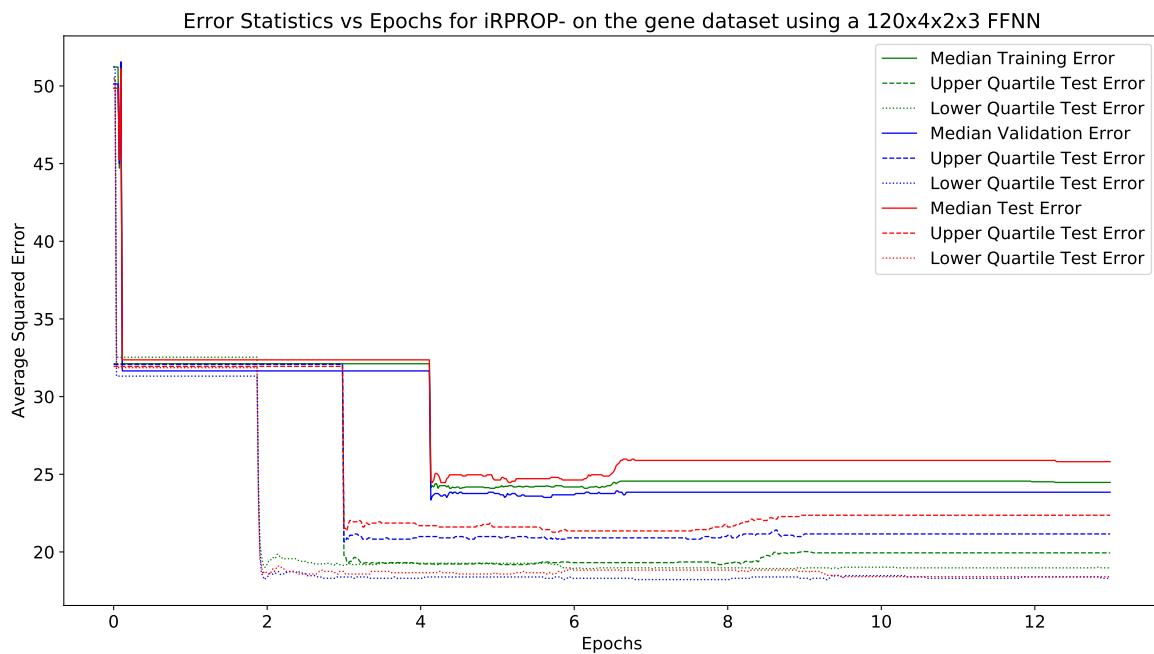
This section displays the results obtained using the gene algorithm.

#### 120 × 4 × 2 × 3 Architecture:

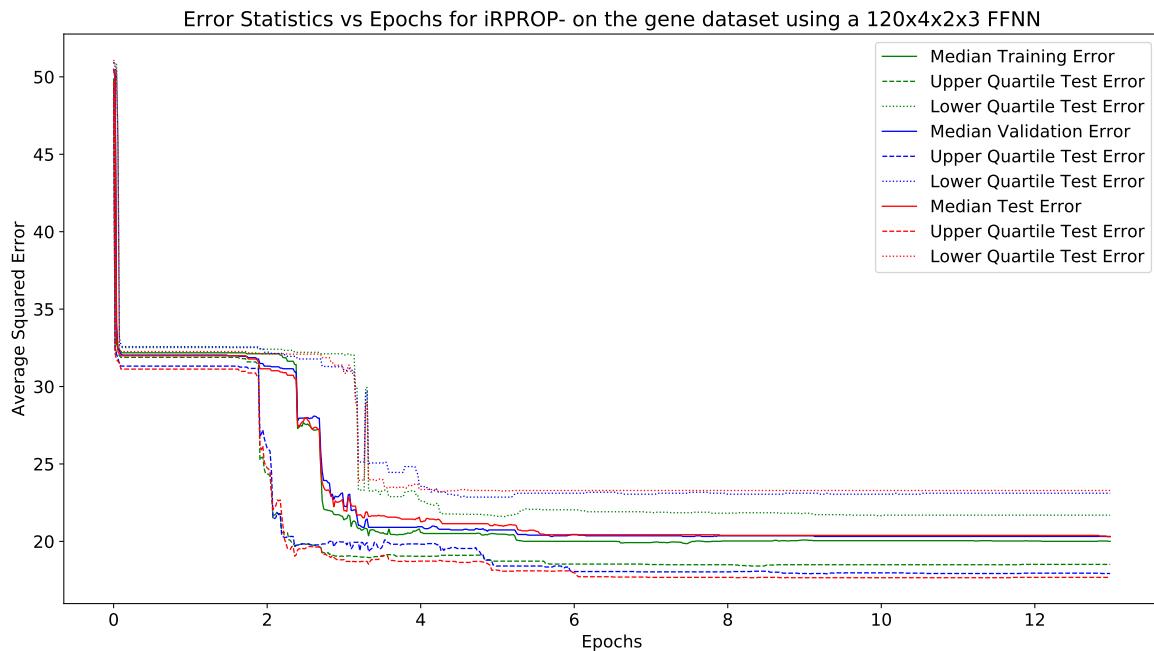
Fig. 311 shows the average and standard deviation of the test, training and validation errors. Fig. 312 shows the median and quartiles of the test, training and validation errors based on the best epoch. Finally, Fig. 313 shows the median and quartiles of the test, training and validation errors per mini-batch.



**Figure 311.** Graph of mean and standard deviation of errors vs epochs



**Figure 312.** Graph of test error vs epochs for the gradient based algorithms



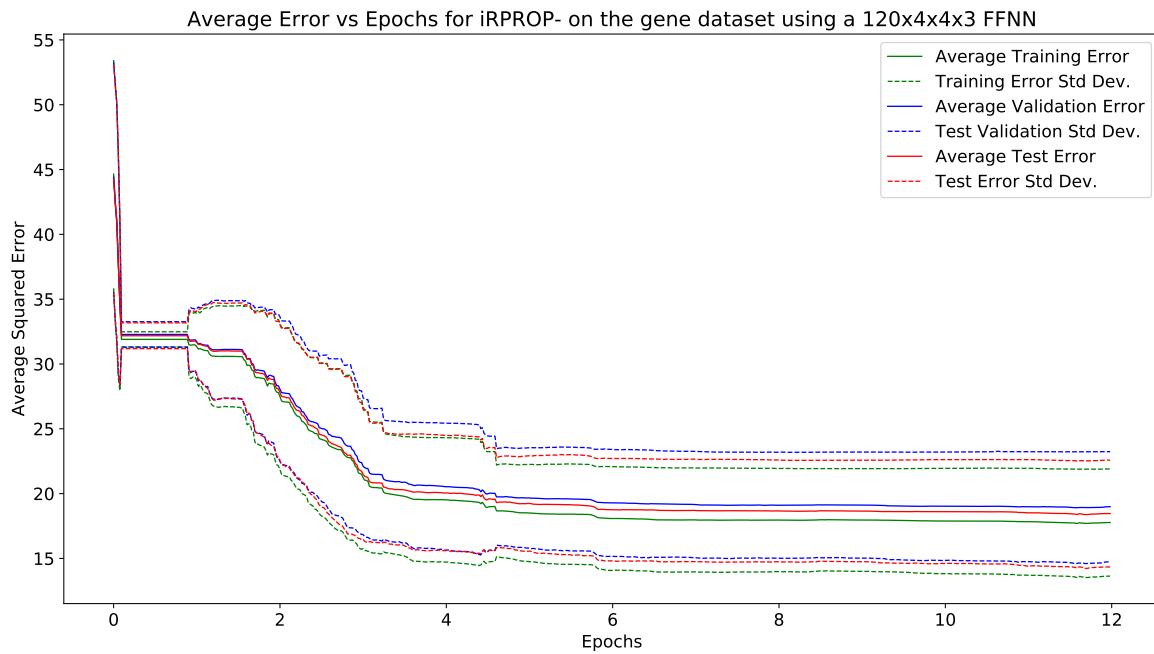
**Figure 313.** Graph of test error vs epochs for the gradient based algorithms

#### 16.4.27 gene

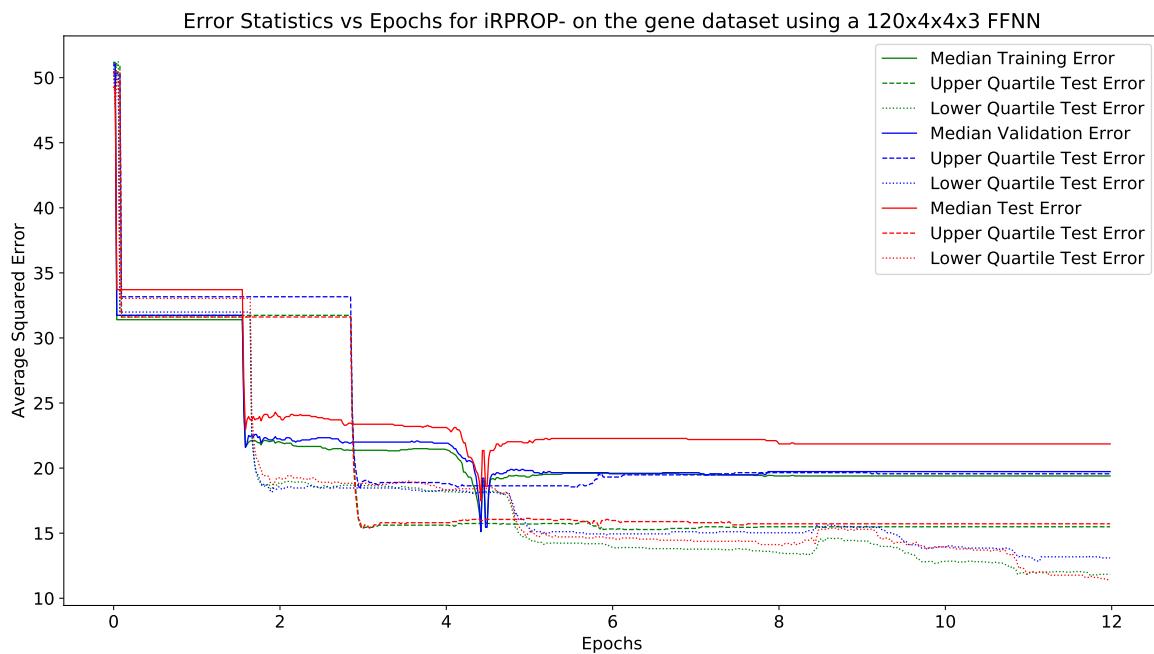
This section displays the results obtained using the gene algorithm.

##### 120 × 4 × 4 × 3 Architecture:

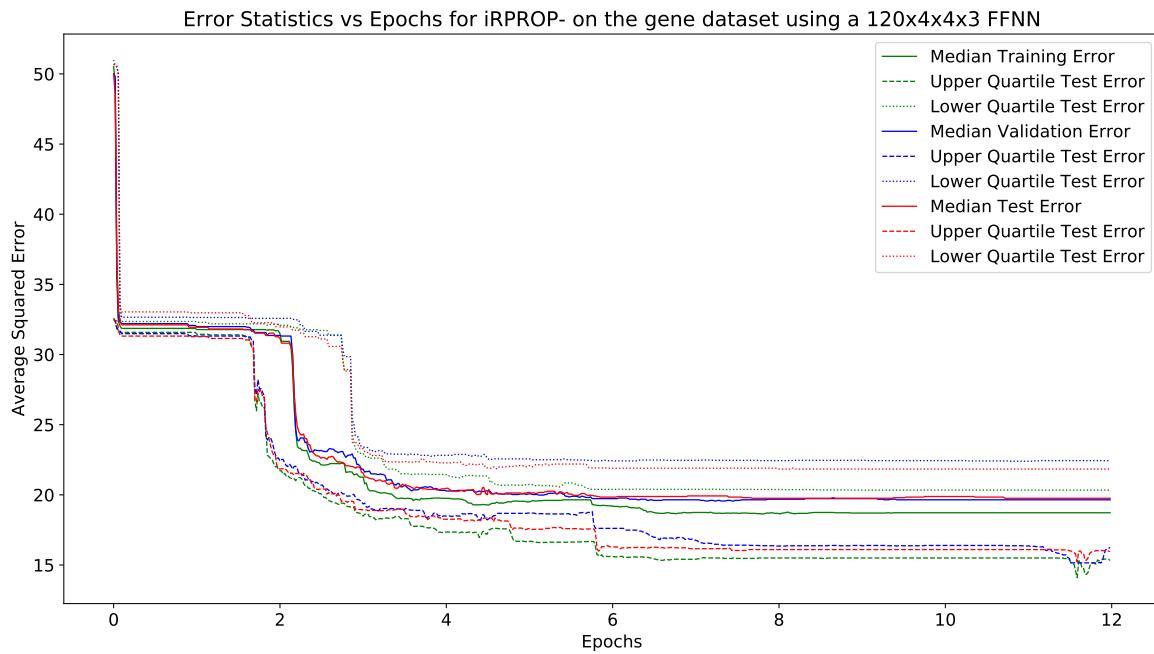
Fig. 314 shows the average and standard deviation of the test, training and validation errors. Fig. 315 shows the median and quartiles of the test, training and validation errors based on the best epoch. Finally, Fig. 316 shows the median and quartiles of the test, training and validation errors per mini-batch.



**Figure 314.** Graph of mean and standard deviation of errors vs epochs



**Figure 315.** Graph of test error vs epochs for the gradient based algorithms



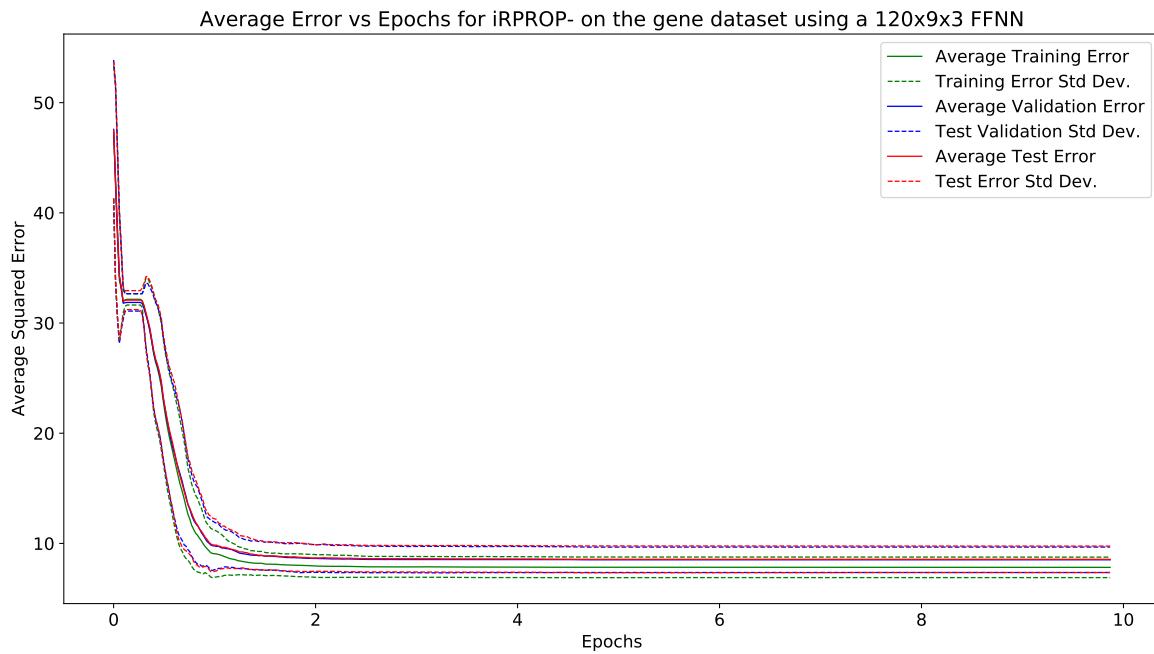
**Figure 316.** Graph of test error vs epochs for the gradient based algorithms

#### 16.4.28 gene

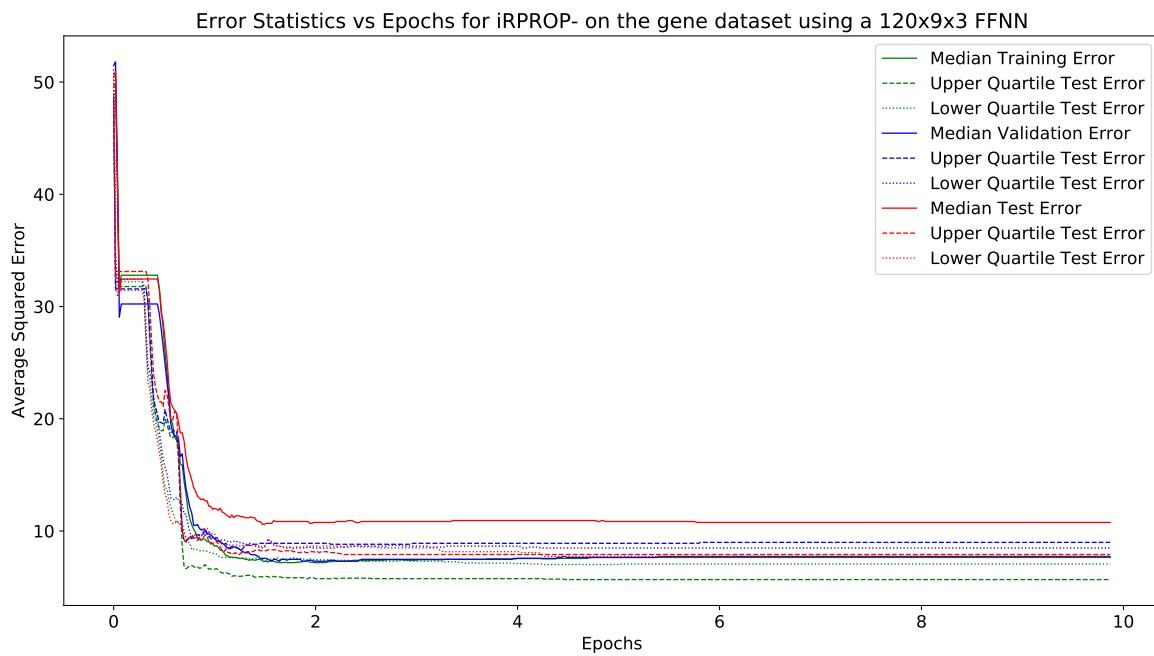
This section displays the results obtained using the gene algorithm.

##### 120 × 9 × 3 Architecture:

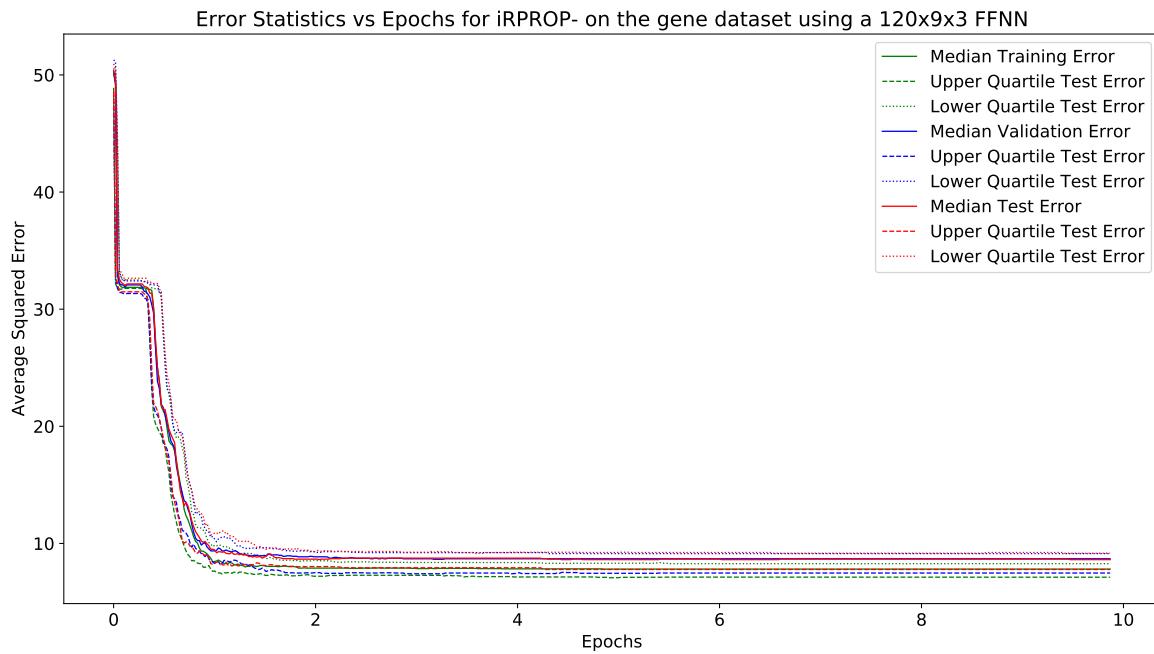
Fig. 317 shows the average and standard deviation of the test, training and validation errors. Fig. 318 shows the median and quartiles of the test, training and validation errors based on the best epoch. Finally, Fig. 319 shows the median and quartiles of the test, training and validation errors per mini-batch.



**Figure 317.** Graph of mean and standard deviation of errors vs epochs



**Figure 318.** Graph of test error vs epochs for the gradient based algorithms



**Figure 319.** Graph of test error vs epochs for the gradient based algorithms

#### 16.4.29 iRPROP+

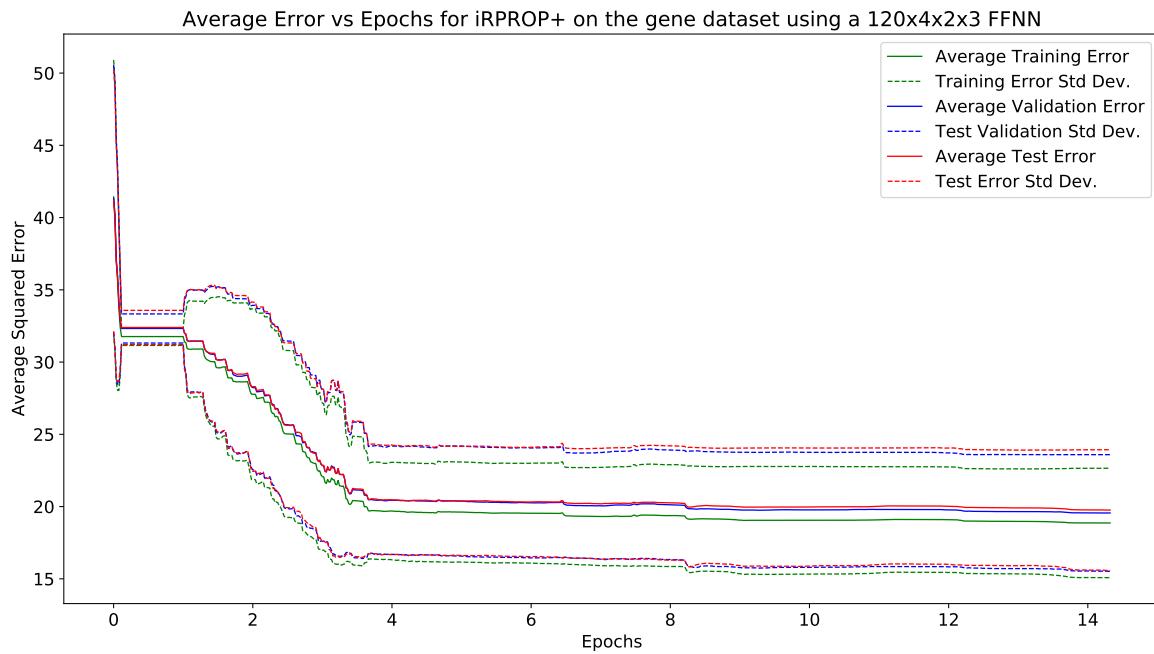
This section displays the results obtained using the iRPROP+ algorithm.

#### 16.4.30 gene

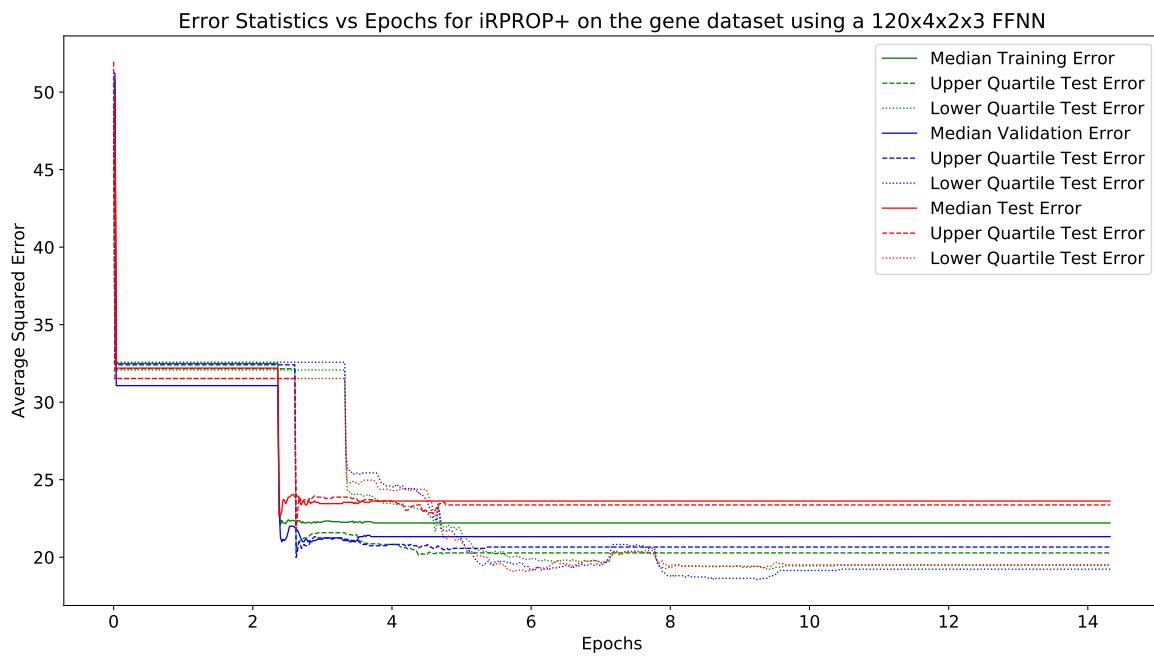
This section displays the results obtained using the gene algorithm.

##### 120 × 4 × 2 × 3 Architecture:

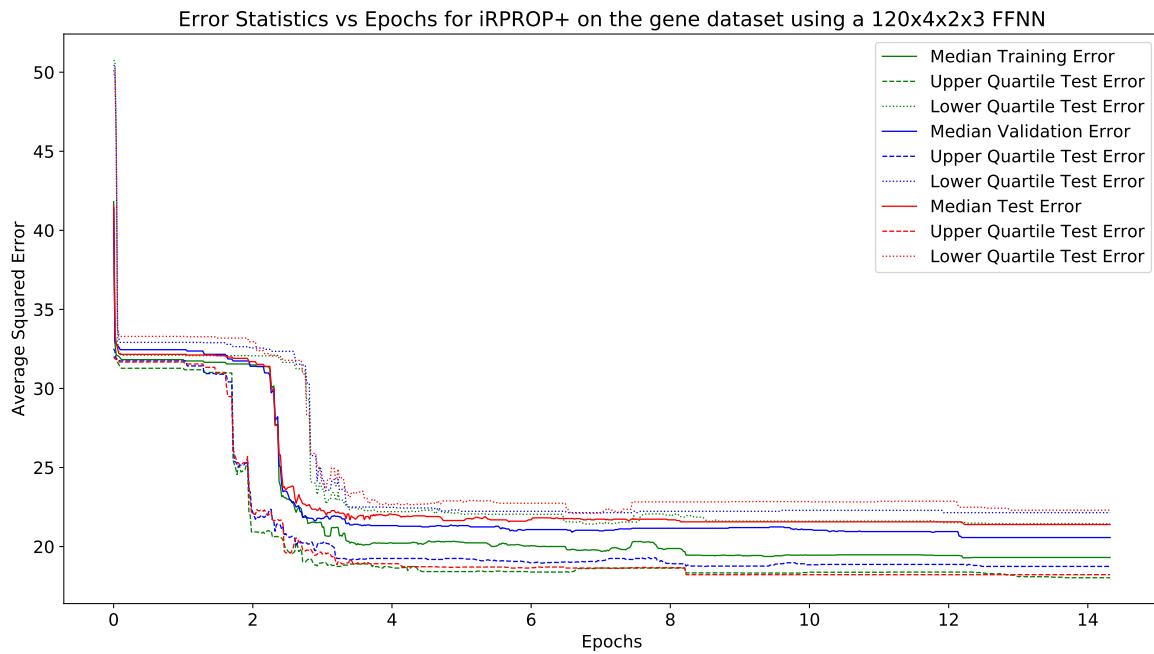
Fig. 320 shows the average and standard deviation of the test, training and validation errors. Fig. 321 shows the median and quartiles of the test, training and validation errors based on the best epoch. Finally, Fig. 322 shows the median and quartiles of the test, training and validation errors per mini-batch.



**Figure 320.** Graph of mean and standard deviation of errors vs epochs



**Figure 321.** Graph of test error vs epochs for the gradient based algorithms



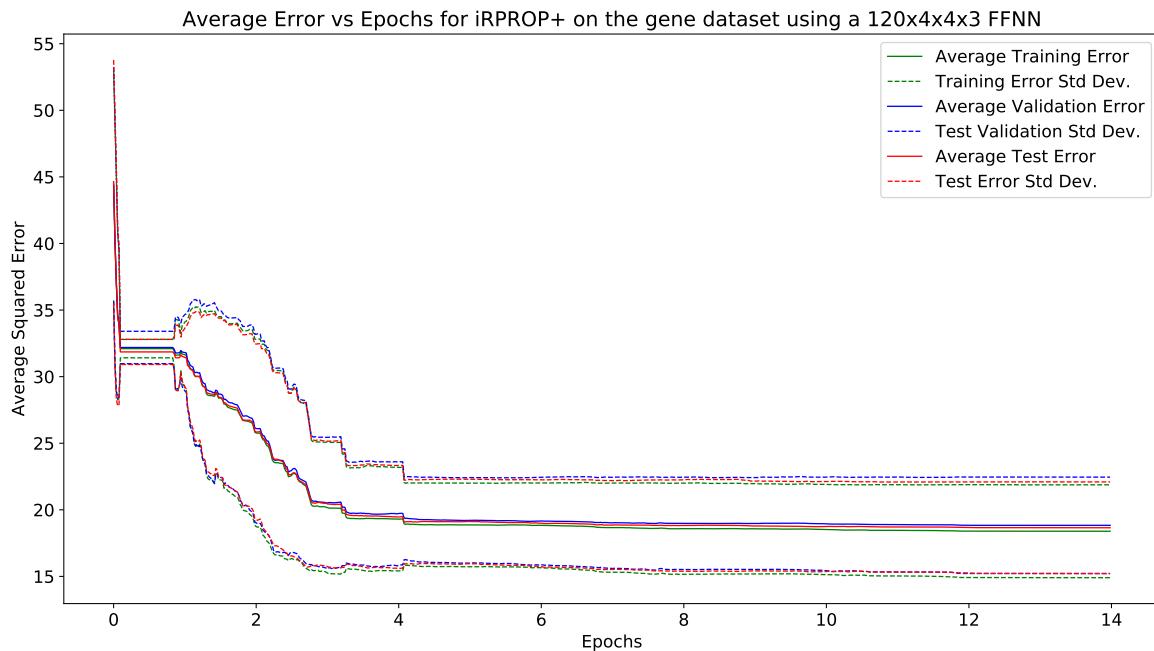
**Figure 322.** Graph of test error vs epochs for the gradient based algorithms

#### 16.4.31 gene

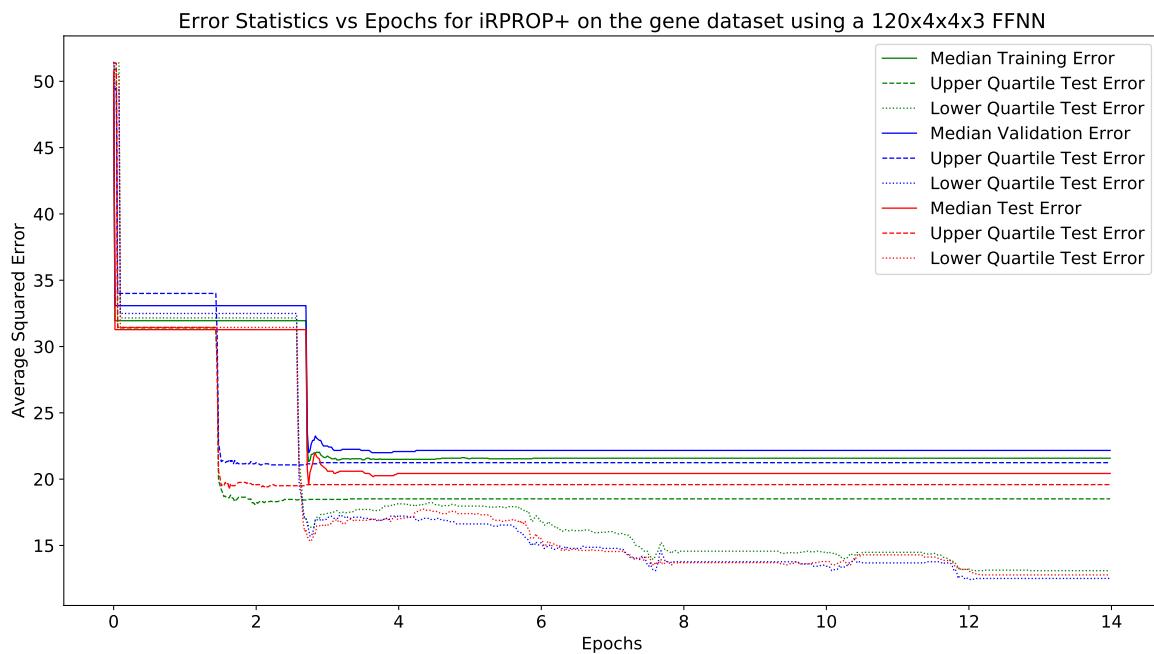
This section displays the results obtained using the gene algorithm.

##### 120 × 4 × 4 × 3 Architecture:

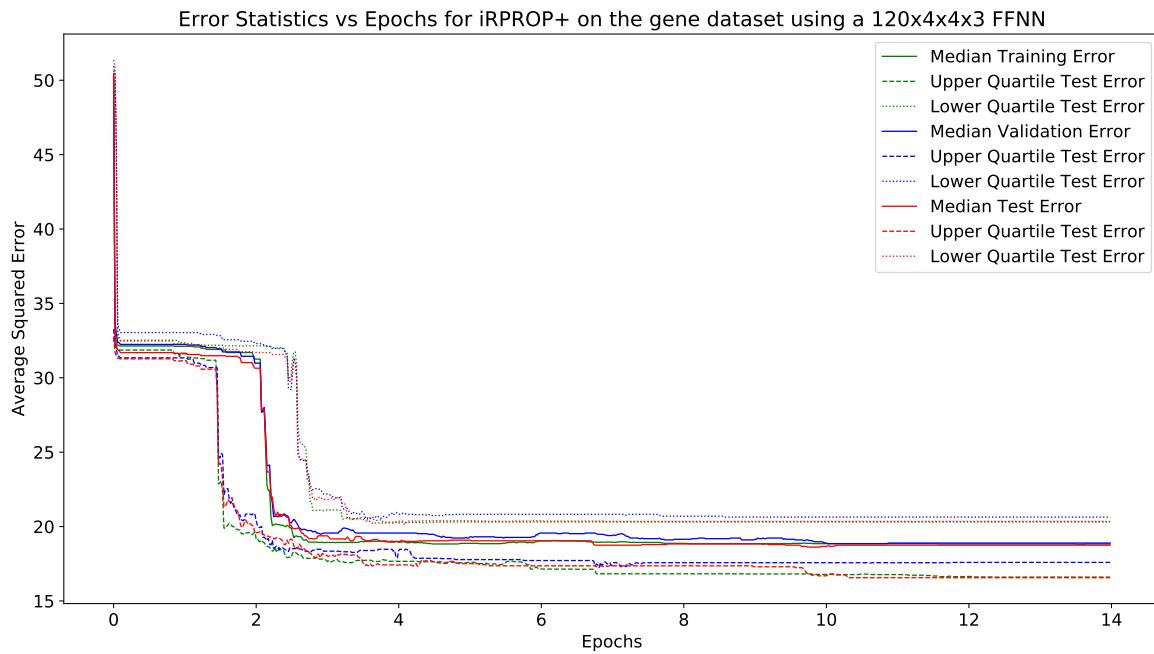
Fig. 323 shows the average and standard deviation of the test, training and validation errors. Fig. 324 shows the median and quartiles of the test, training and validation errors based on the best epoch. Finally, Fig. 325 shows the median and quartiles of the test, training and validation errors per mini-batch.



**Figure 323.** Graph of mean and standard deviation of errors vs epochs



**Figure 324.** Graph of test error vs epochs for the gradient based algorithms



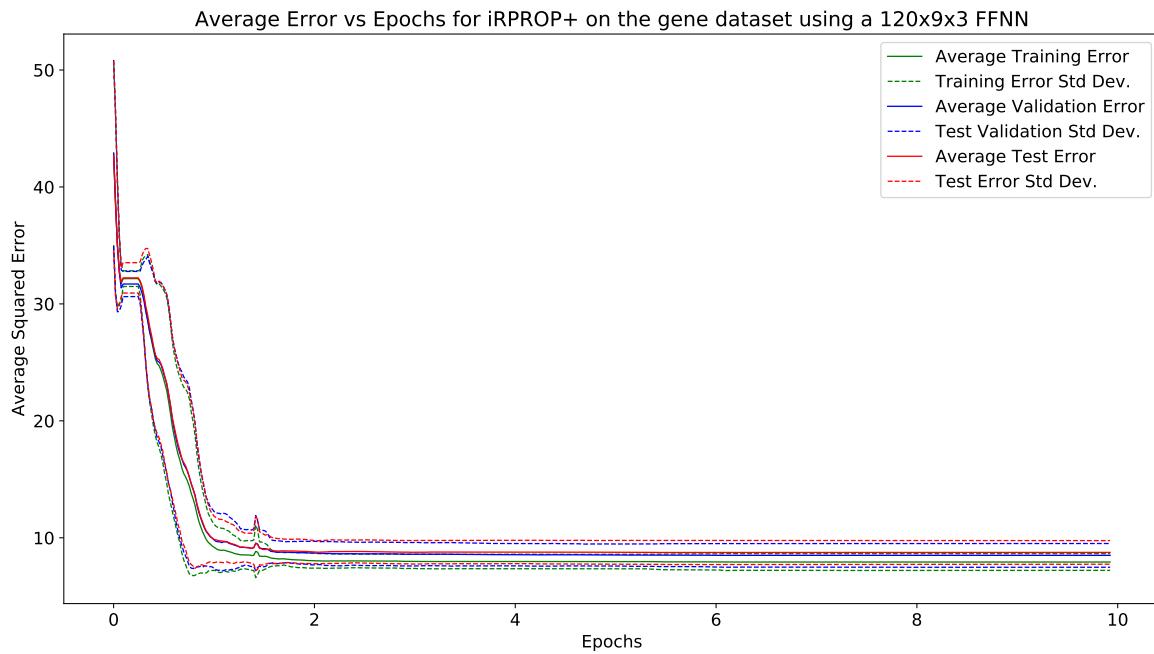
**Figure 325.** Graph of test error vs epochs for the gradient based algorithms

### 16.4.32 gene

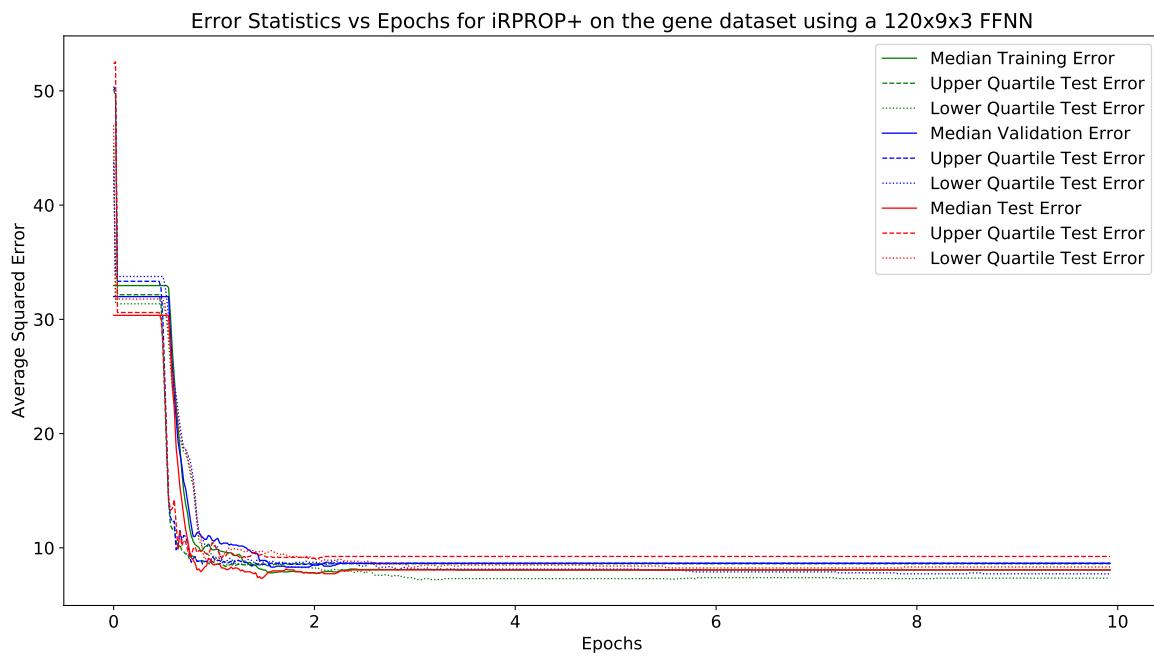
This section displays the results obtained using the gene algorithm.

#### 120 × 9 × 3 Architecture:

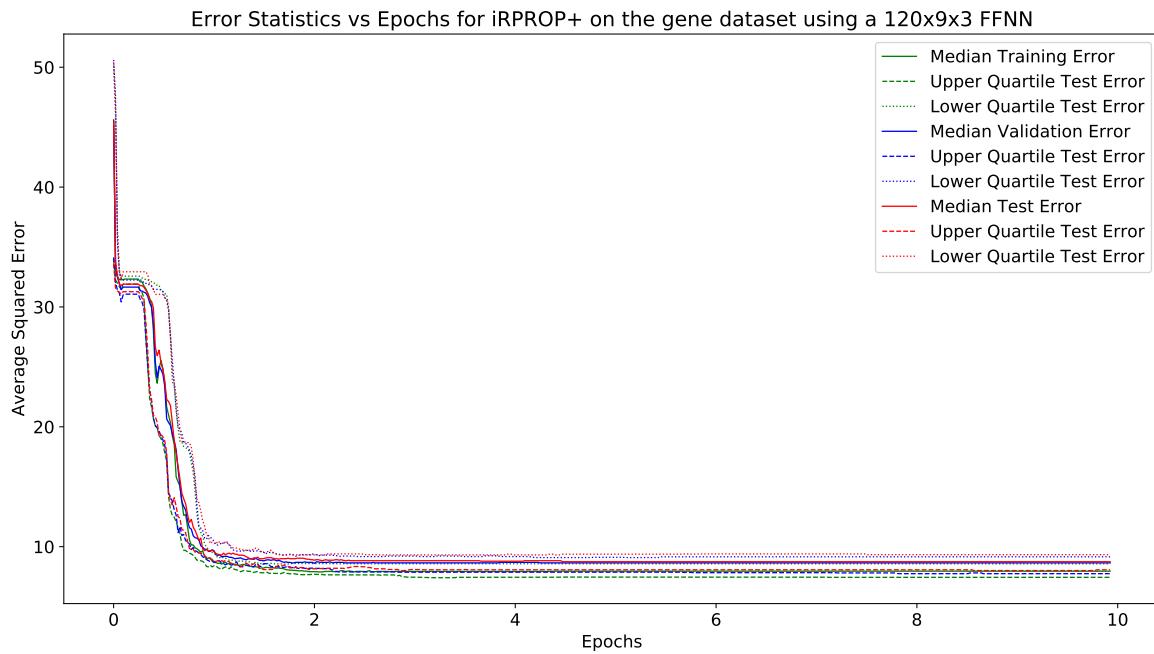
Fig. 326 shows the average and standard deviation of the test, training and validation errors. Fig. 327 shows the median and quartiles of the test, training and validation errors based on the best epoch. Finally, Fig. 328 shows the median and quartiles of the test, training and validation errors per mini-batch.



**Figure 326.** Graph of mean and standard deviation of errors vs epochs



**Figure 327.** Graph of test error vs epochs for the gradient based algorithms



**Figure 328.** Graph of test error vs epochs for the gradient based algorithms

#### 16.4.33 PSO

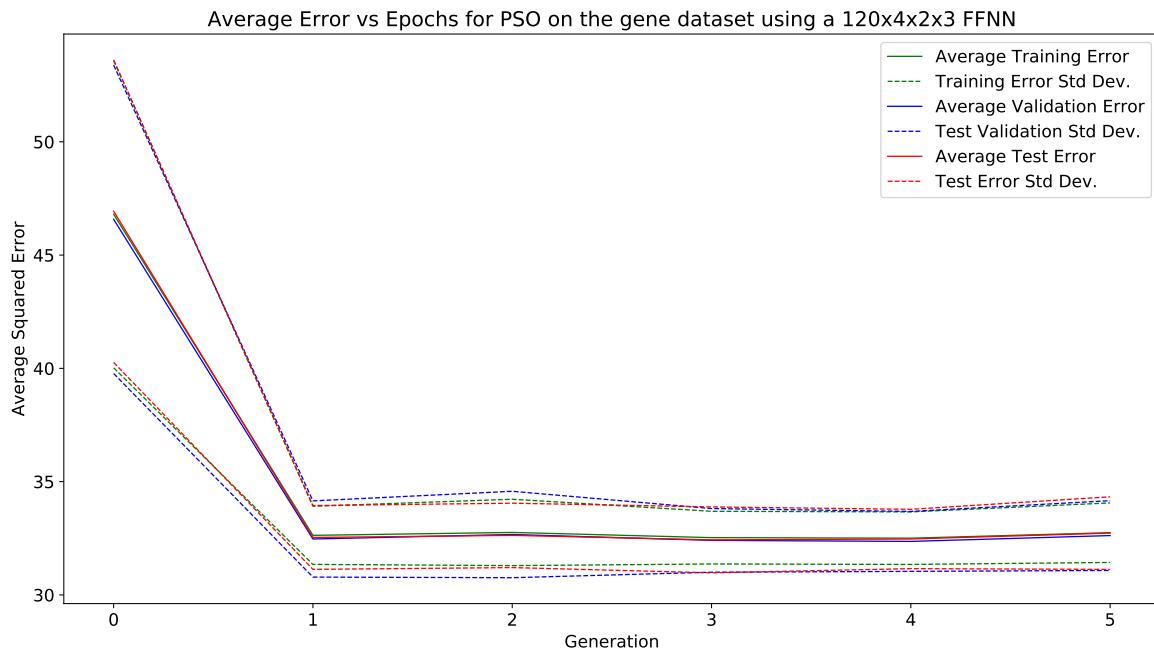
This section displays the results obtained using the PSO algorithm.

#### 16.4.34 gene

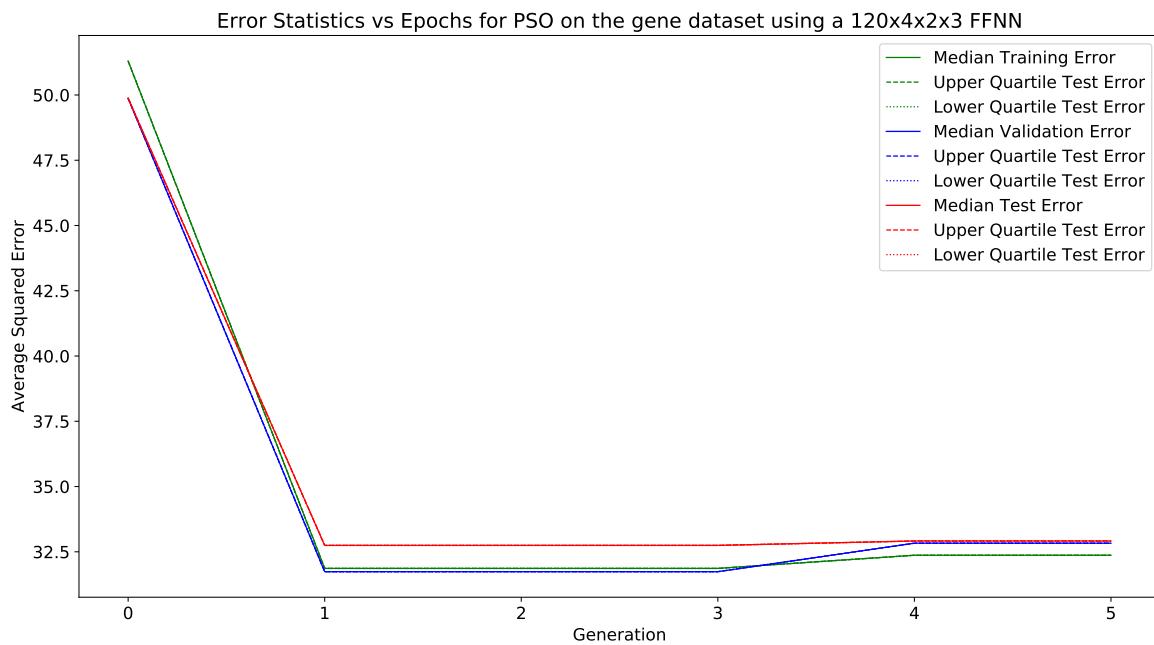
This section displays the results obtained using the gene algorithm.

##### 120 × 4 × 2 × 3 Architecture:

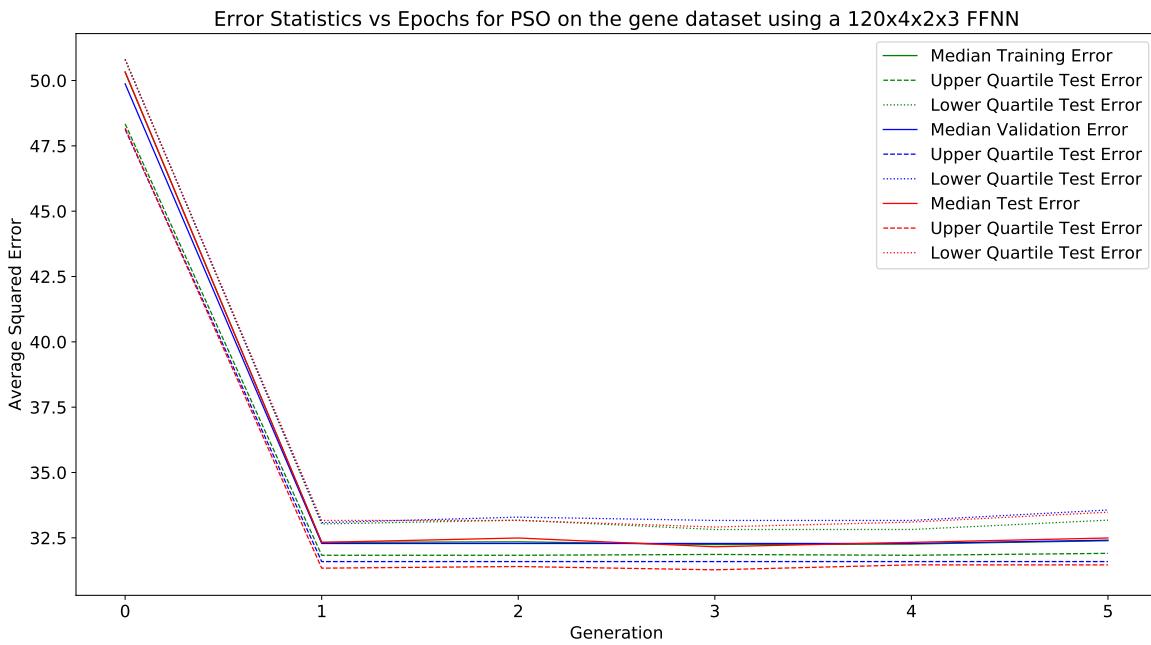
Fig. 329 shows the average and standard deviation of the test, training and validation errors. Fig. 330 shows the median and quartiles of the test, training and validation errors based on the best epoch. Finally, Fig. 331 shows the median and quartiles of the test, training and validation errors per mini-batch.



**Figure 329.** Graph of mean and standard deviation of errors vs epochs



**Figure 330.** Graph of test error vs epochs for the gradient based algorithms



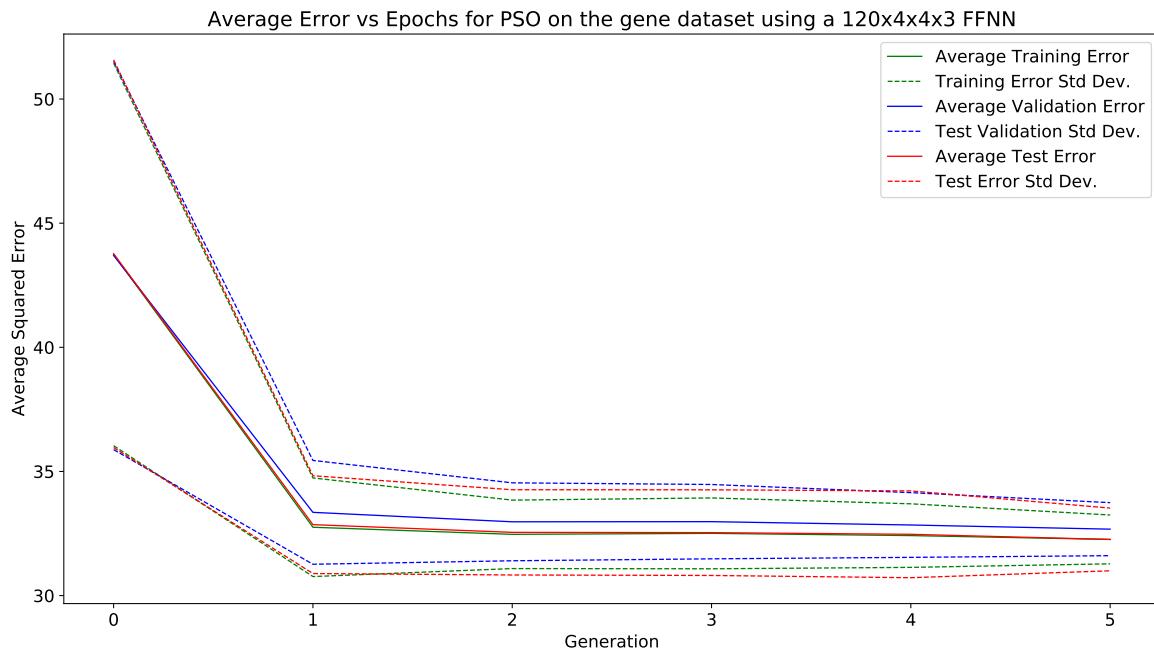
**Figure 331.** Graph of test error vs epochs for the gradient based algorithms

#### 16.4.35 gene

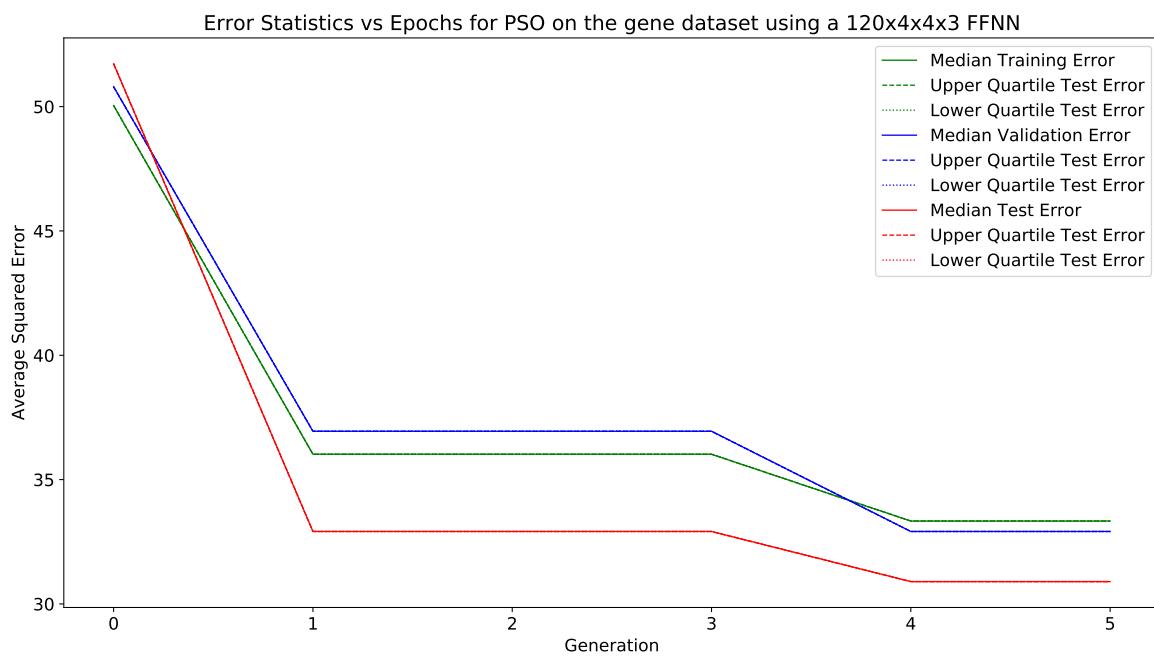
This section displays the results obtained using the gene algorithm.

##### 120 × 4 × 4 × 3 Architecture:

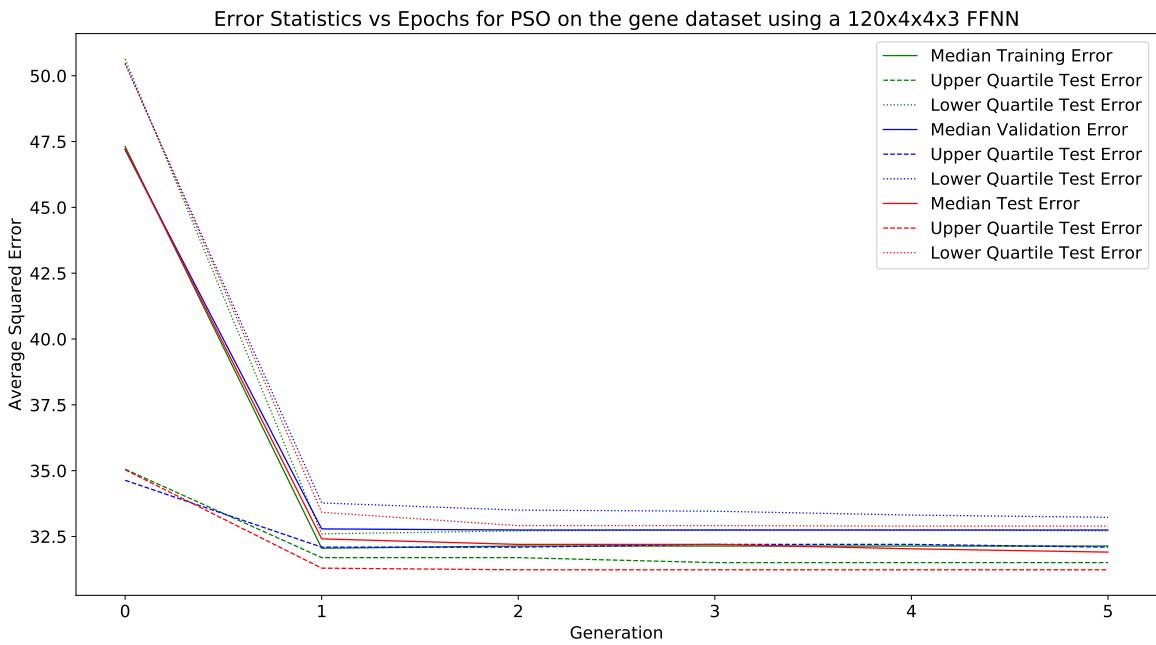
Fig. 332 shows the average and standard deviation of the test, training and validation errors. Fig. 333 shows the median and quartiles of the test, training and validation errors based on the best epoch. Finally, Fig. 334 shows the median and quartiles of the test, training and validation errors per mini-batch.



**Figure 332.** Graph of mean and standard deviation of errors vs epochs



**Figure 333.** Graph of test error vs epochs for the gradient based algorithms



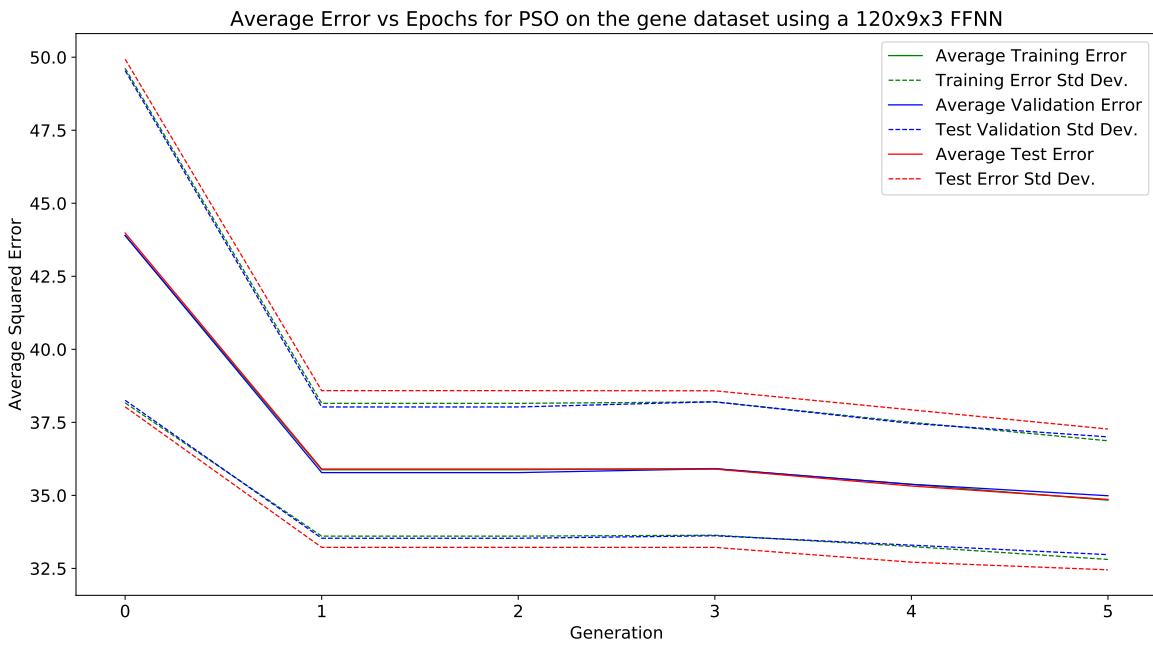
**Figure 334.** Graph of test error vs epochs for the gradient based algorithms

#### 16.4.36 gene

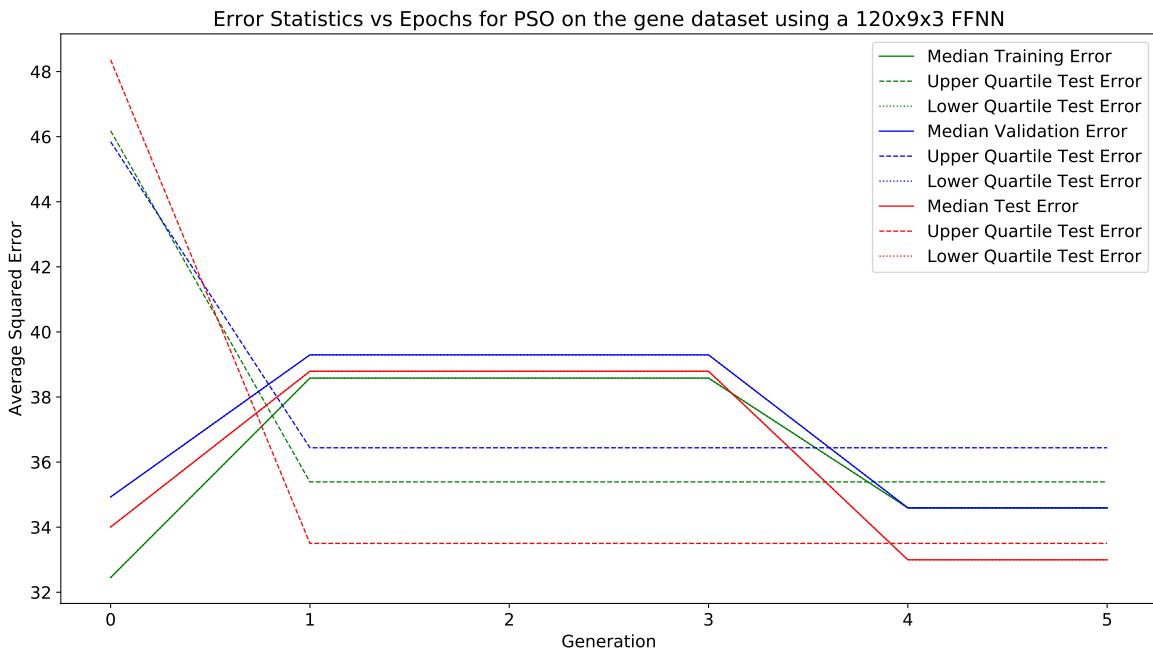
This section displays the results obtained using the gene algorithm.

##### 120 × 9 × 3 Architecture:

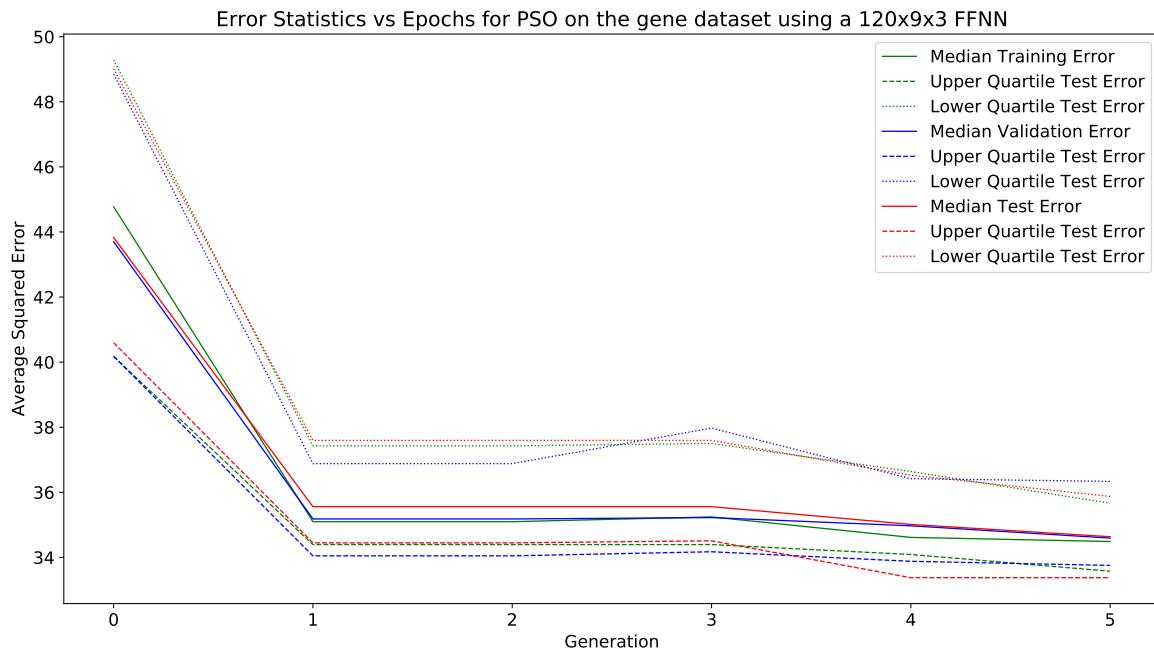
Fig. 335 shows the average and standard deviation of the test, training and validation errors. Fig. 336 shows the median and quartiles of the test, training and validation errors based on the best epoch. Finally, Fig. 337 shows the median and quartiles of the test, training and validation errors per mini-batch.



**Figure 335.** Graph of mean and standard deviation of errors vs epochs



**Figure 336.** Graph of test error vs epochs for the gradient based algorithms



**Figure 337.** Graph of test error vs epochs for the gradient based algorithms

#### 16.4.37 QuickProp

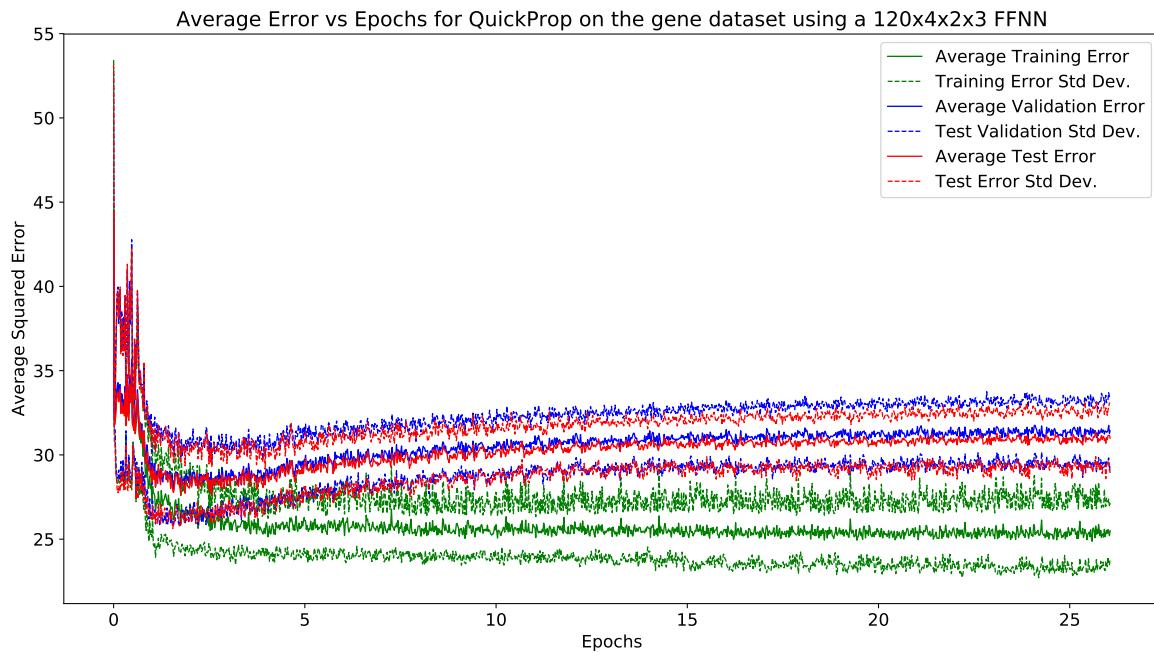
This section displays the results obtained using the QuickProp algorithm.

#### 16.4.38 gene

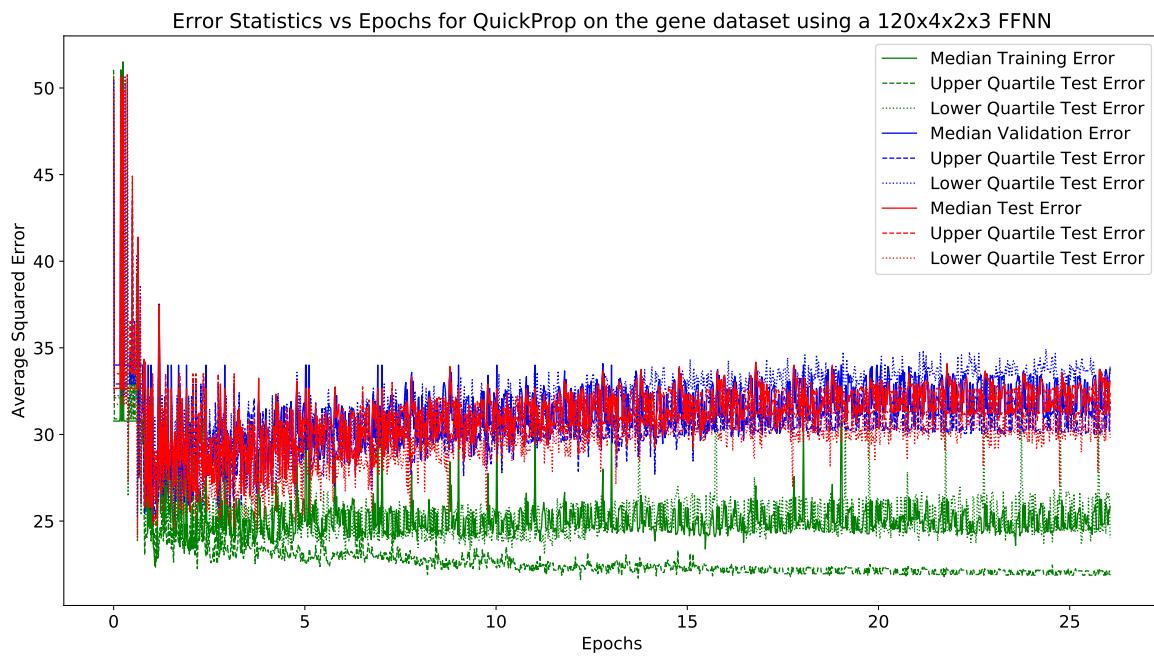
This section displays the results obtained using the gene algorithm.

##### 120 × 4 × 2 × 3 Architecture:

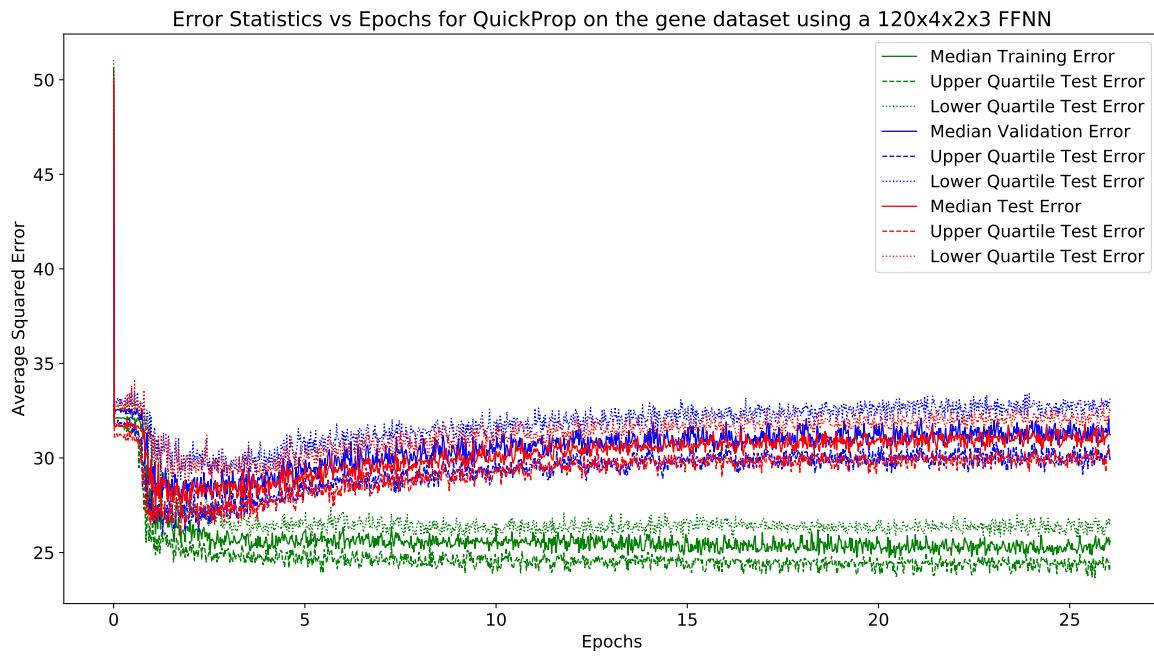
Fig. 338 shows the average and standard deviation of the test, training and validation errors. Fig. 339 shows the median and quartiles of the test, training and validation errors based on the best epoch. Finally, Fig. 340 shows the median and quartiles of the test, training and validation errors per mini-batch.



**Figure 338.** Graph of mean and standard deviation of errors vs epochs



**Figure 339.** Graph of test error vs epochs for the gradient based algorithms



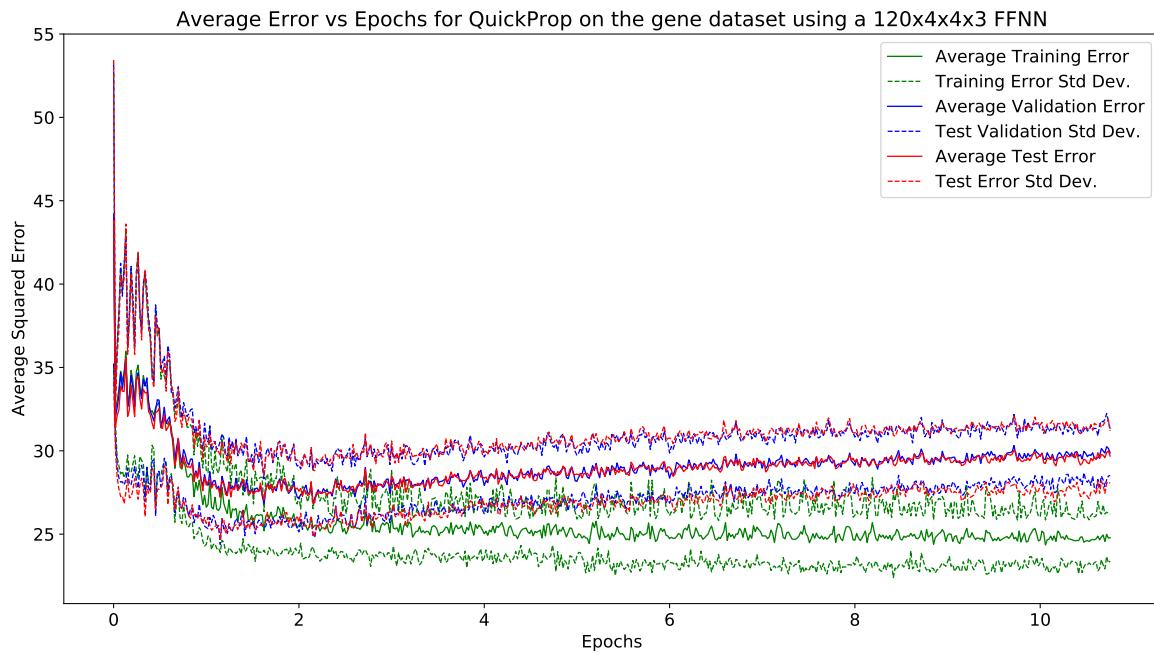
**Figure 340.** Graph of test error vs epochs for the gradient based algorithms

#### 16.4.39 gene

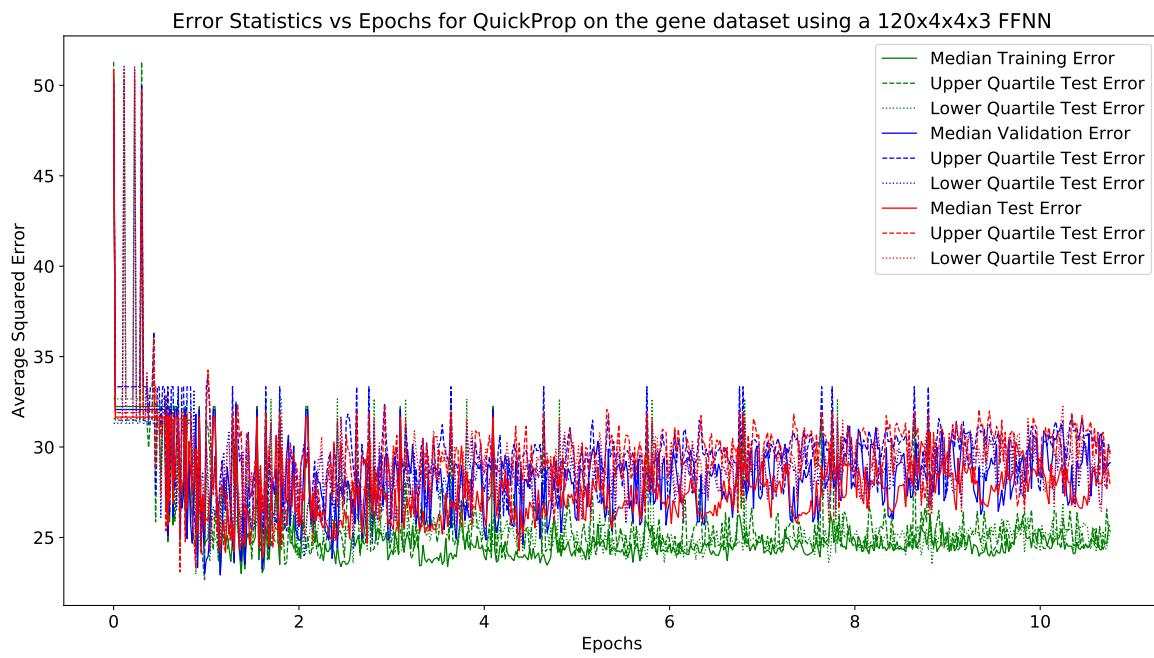
This section displays the results obtained using the gene algorithm.

##### 120 × 4 × 4 × 3 Architecture:

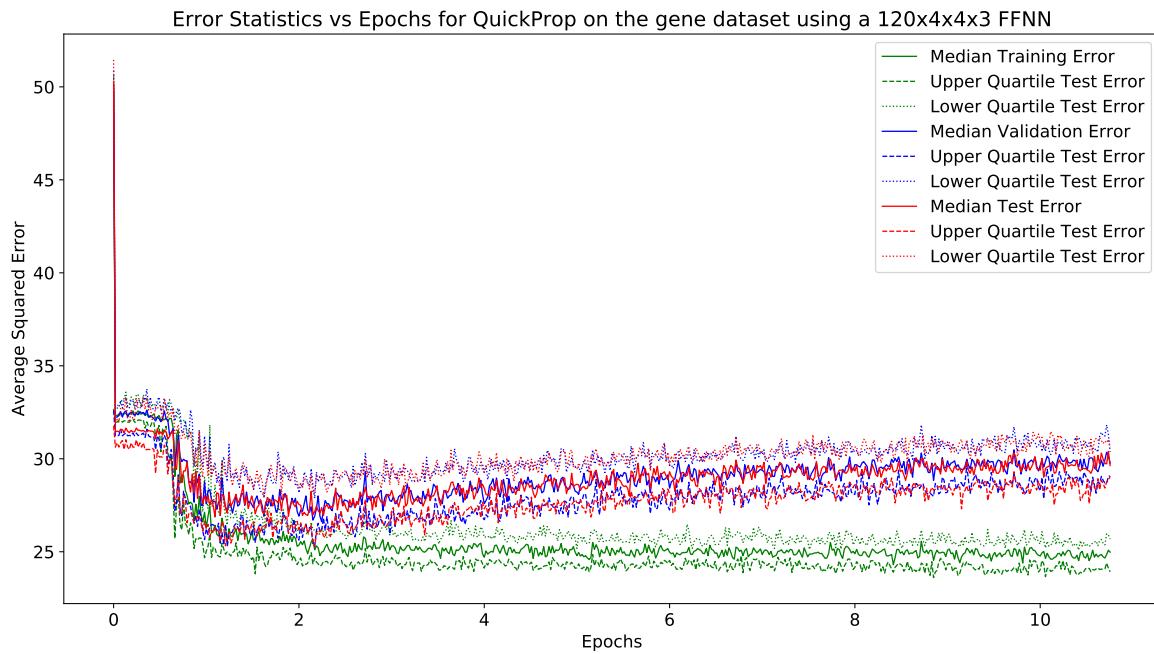
Fig. 341 shows the average and standard deviation of the test, training and validation errors. Fig. 342 shows the median and quartiles of the test, training and validation errors based on the best epoch. Finally, Fig. 343 shows the median and quartiles of the test, training and validation errors per mini-batch.



**Figure 341.** Graph of mean and standard deviation of errors vs epochs



**Figure 342.** Graph of test error vs epochs for the gradient based algorithms



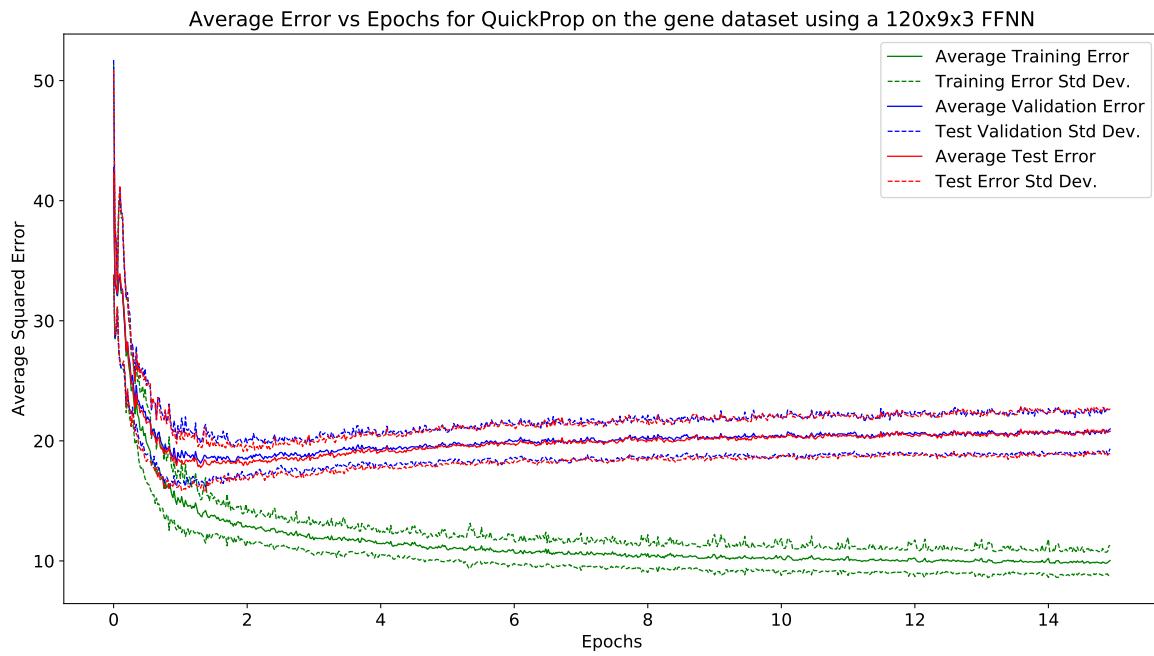
**Figure 343.** Graph of test error vs epochs for the gradient based algorithms

#### 16.4.40 gene

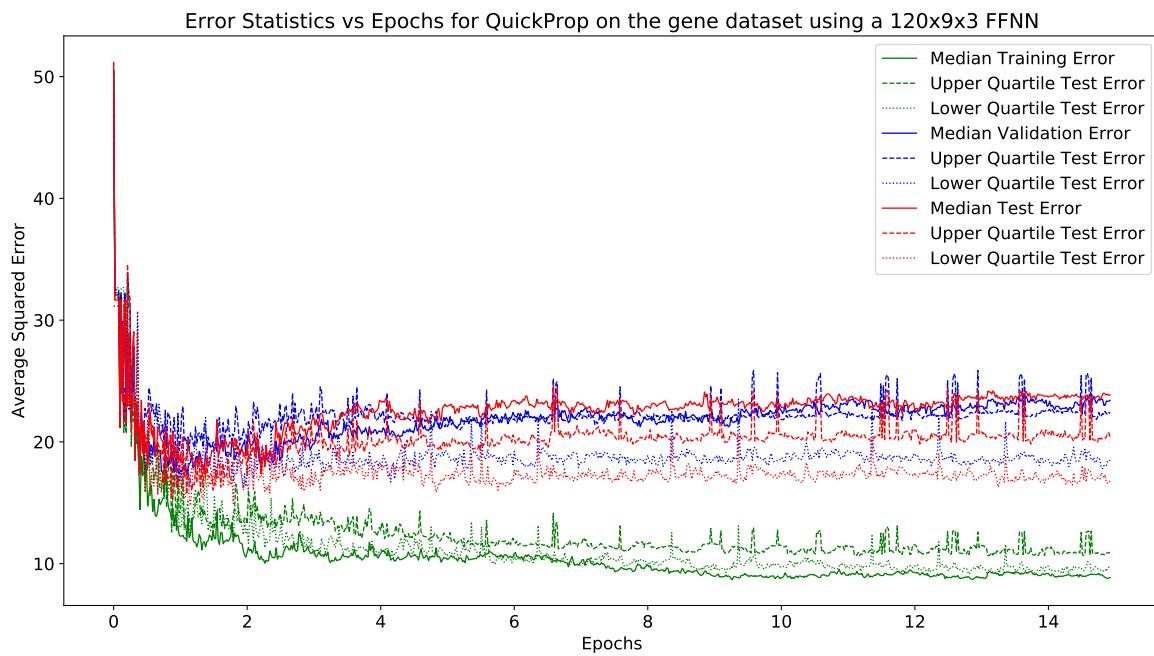
This section displays the results obtained using the gene algorithm.

##### 120 × 9 × 3 Architecture:

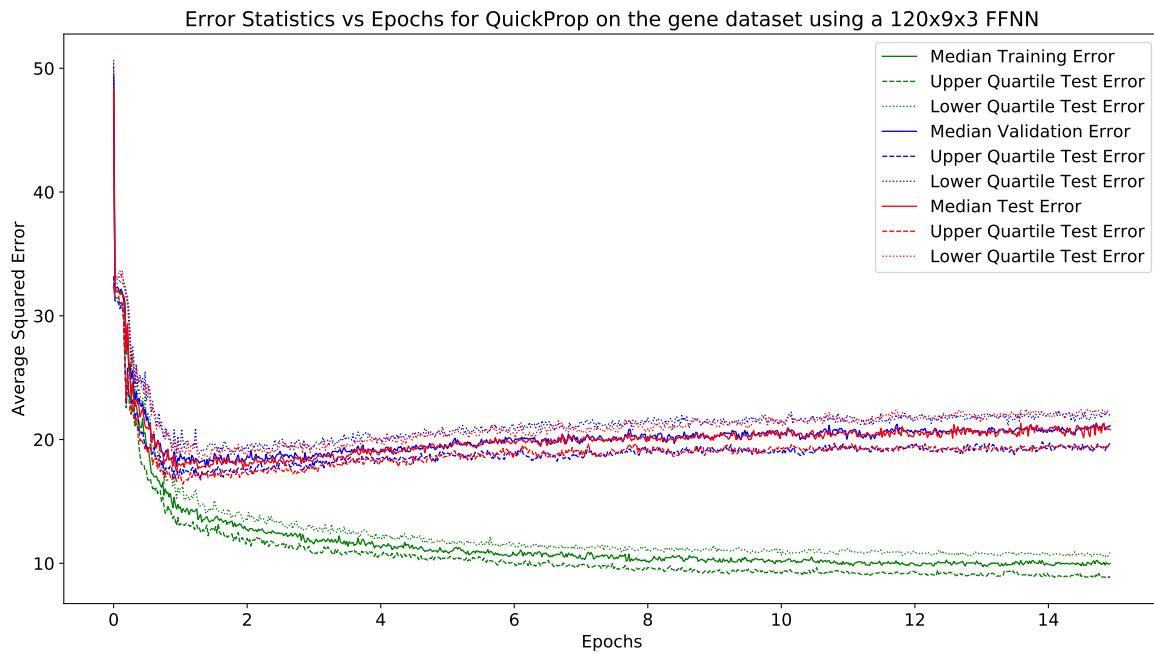
Fig. 344 shows the average and standard deviation of the test, training and validation errors. Fig. 345 shows the median and quartiles of the test, training and validation errors based on the best epoch. Finally, Fig. 346 shows the median and quartiles of the test, training and validation errors per mini-batch.



**Figure 344.** Graph of mean and standard deviation of errors vs epochs



**Figure 345.** Graph of test error vs epochs for the gradient based algorithms



**Figure 346.** Graph of test error vs epochs for the gradient based algorithms

#### 16.4.41 RPROP-

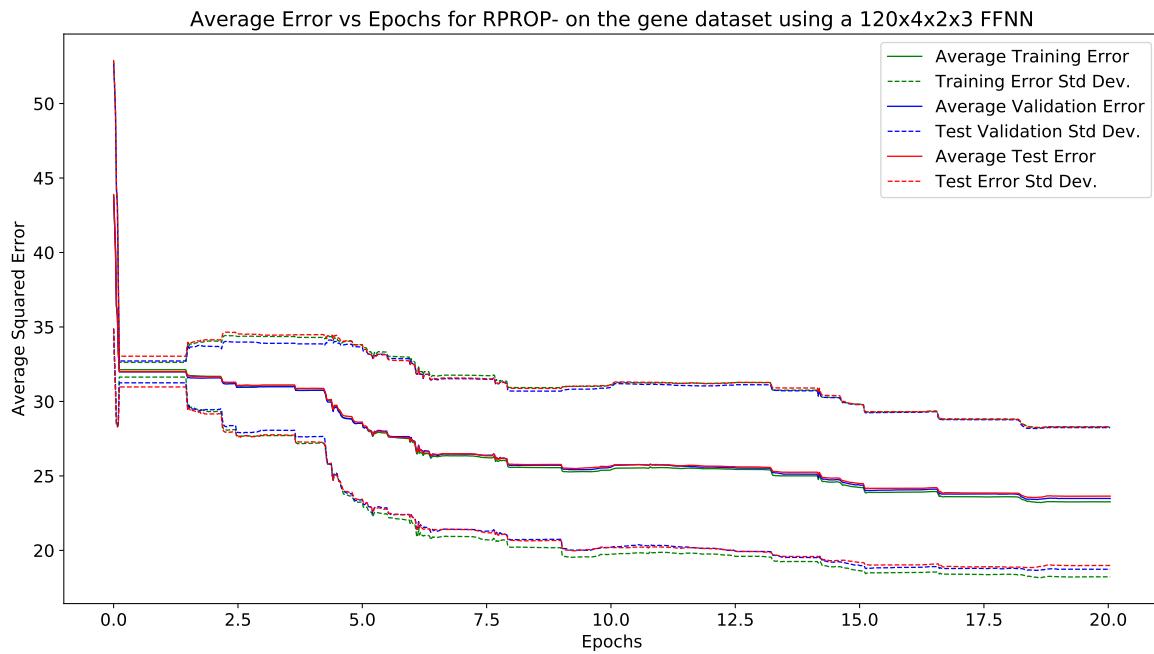
This section displays the results obtained using the RPROP- algorithm.

#### 16.4.42 gene

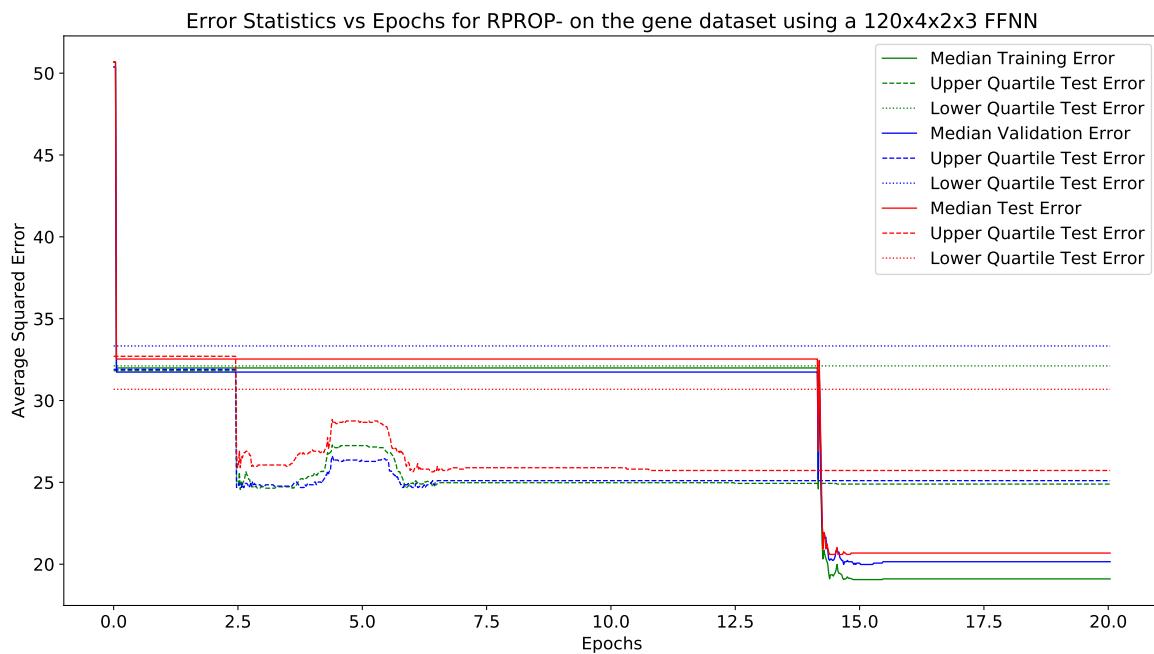
This section displays the results obtained using the gene algorithm.

##### 120 × 4 × 2 × 3 Architecture:

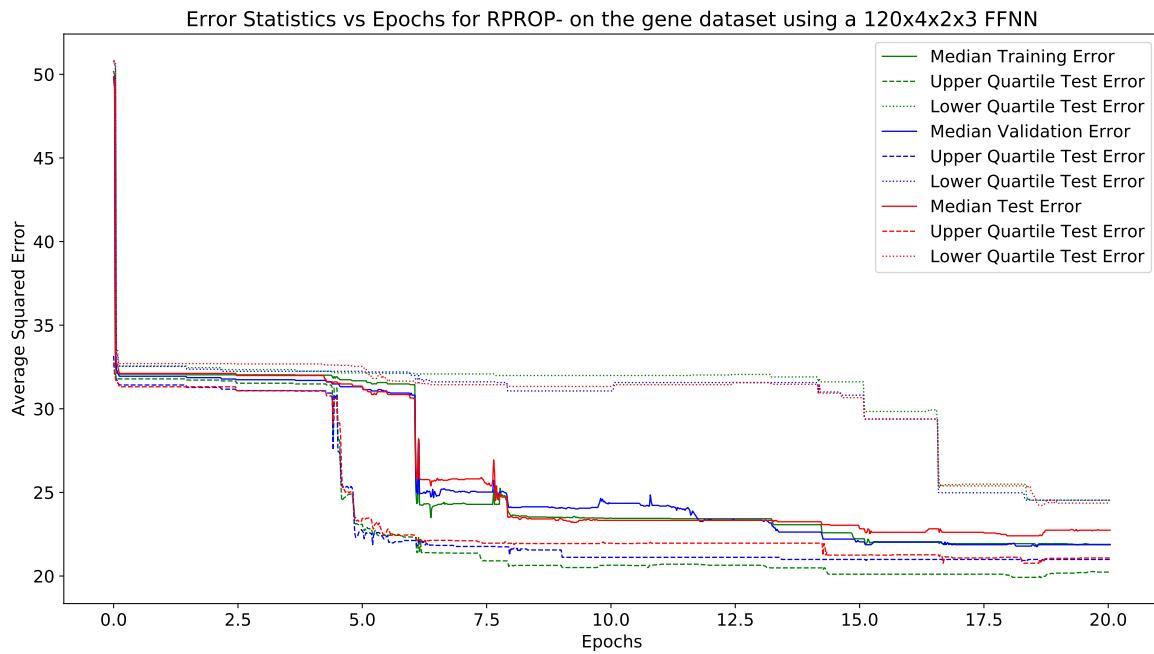
Fig. 347 shows the average and standard deviation of the test, training and validation errors. Fig. 348 shows the median and quartiles of the test, training and validation errors based on the best epoch. Finally, Fig. 349 shows the median and quartiles of the test, training and validation errors per mini-batch.



**Figure 347.** Graph of mean and standard deviation of errors vs epochs



**Figure 348.** Graph of test error vs epochs for the gradient based algorithms



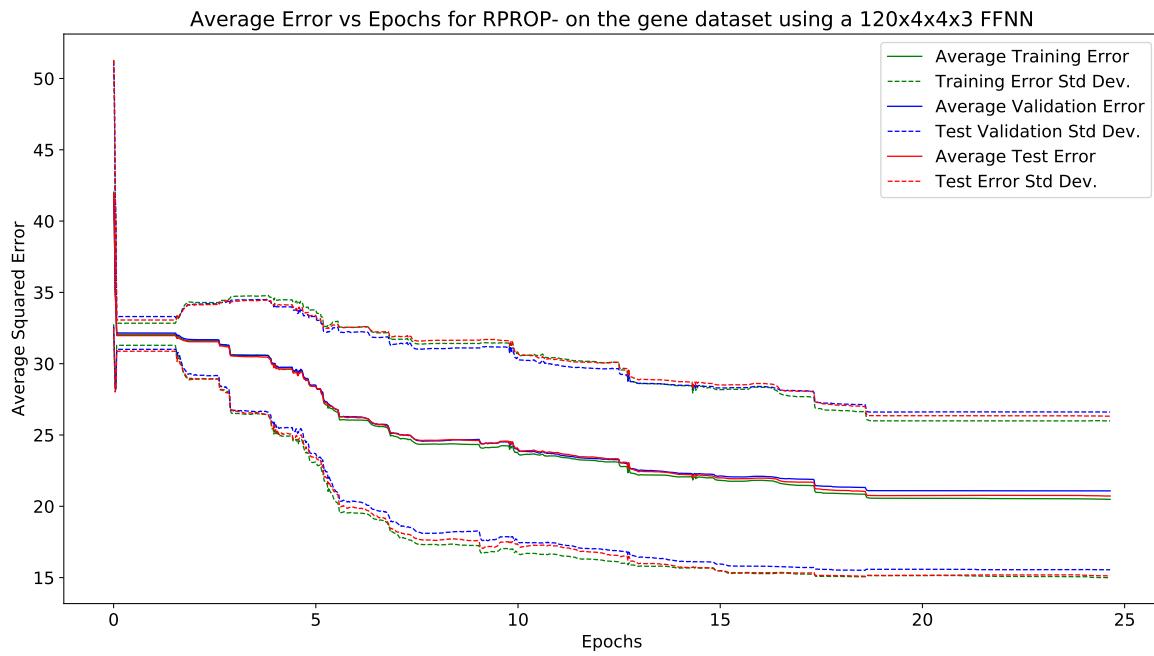
**Figure 349.** Graph of test error vs epochs for the gradient based algorithms

#### 16.4.43 gene

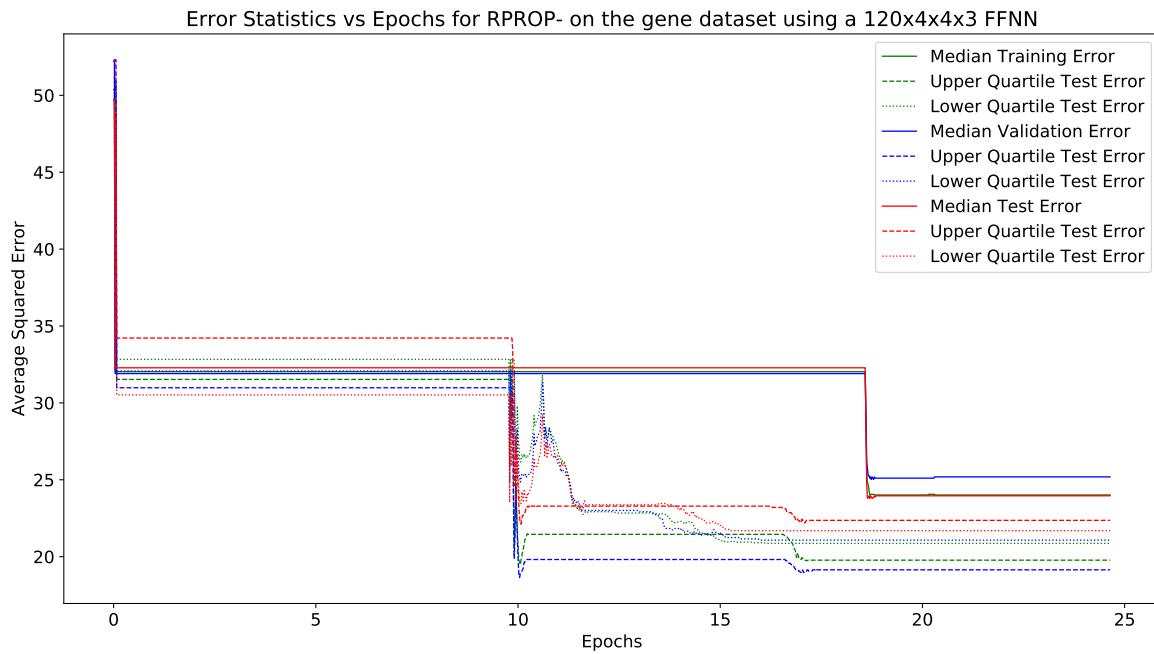
This section displays the results obtained using the gene algorithm.

##### 120 × 4 × 4 × 3 Architecture:

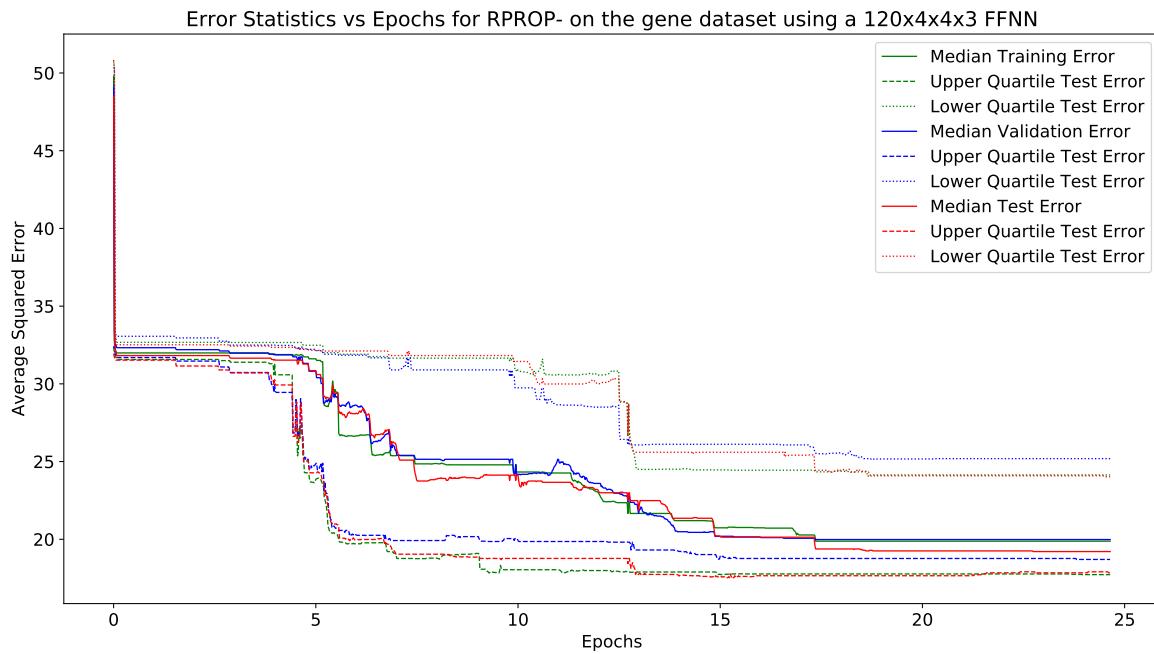
Fig. 350 shows the average and standard deviation of the test, training and validation errors. Fig. 351 shows the median and quartiles of the test, training and validation errors based on the best epoch. Finally, Fig. 352 shows the median and quartiles of the test, training and validation errors per mini-batch.



**Figure 350.** Graph of mean and standard deviation of errors vs epochs



**Figure 351.** Graph of test error vs epochs for the gradient based algorithms



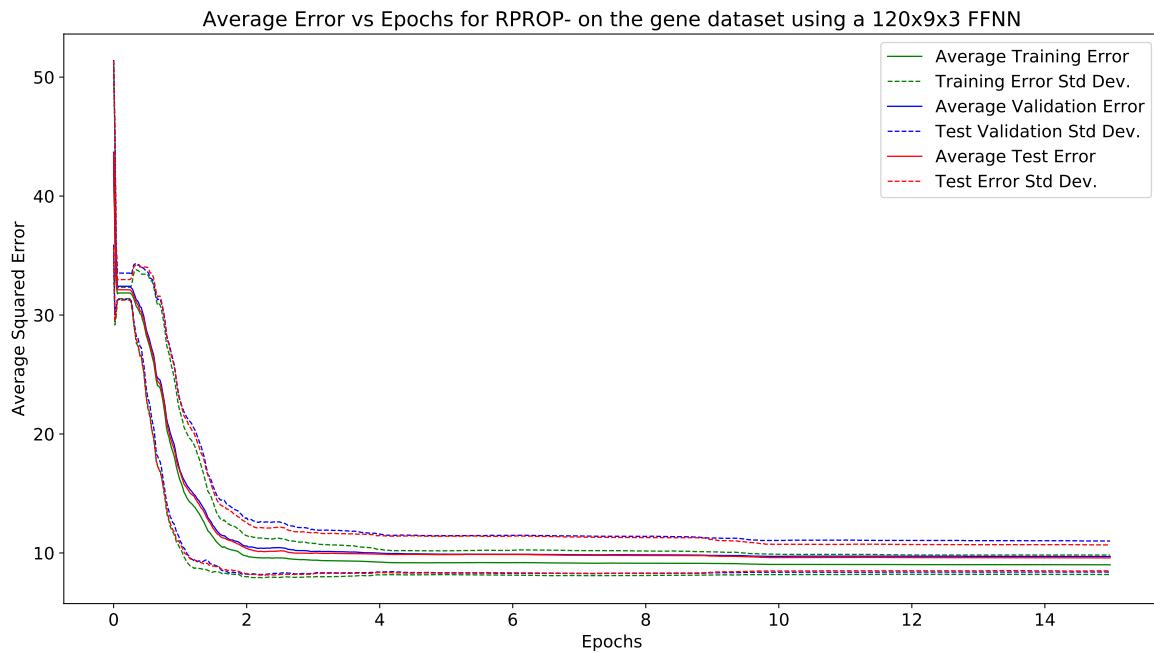
**Figure 352.** Graph of test error vs epochs for the gradient based algorithms

#### 16.4.44 gene

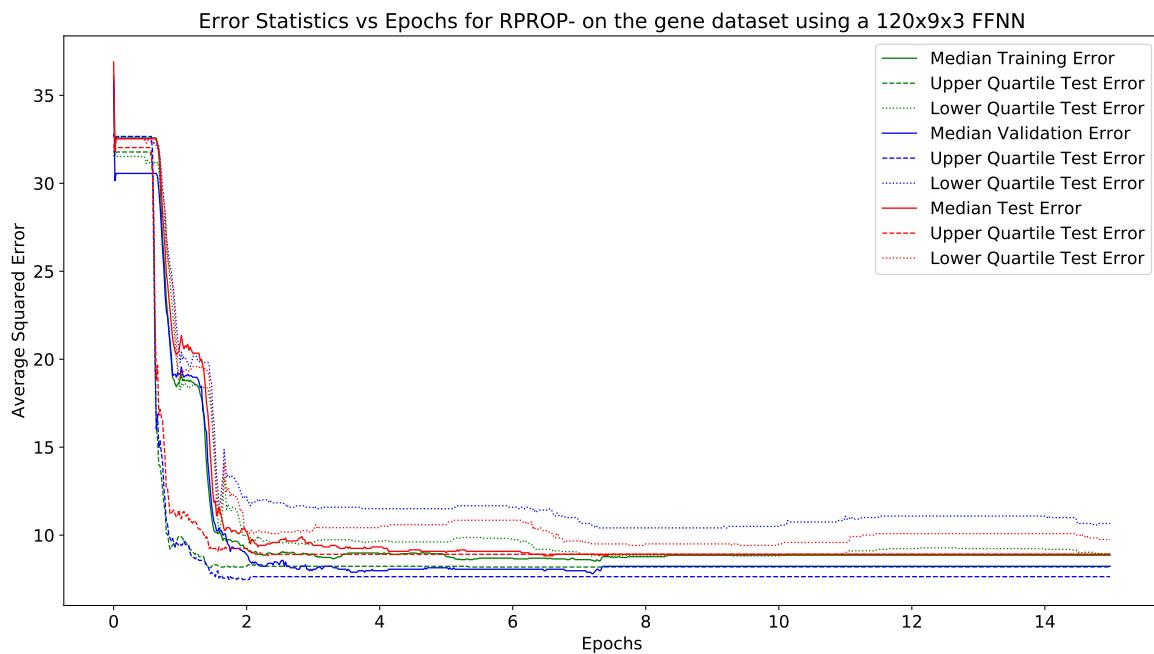
This section displays the results obtained using the gene algorithm.

##### 120 × 9 × 3 Architecture:

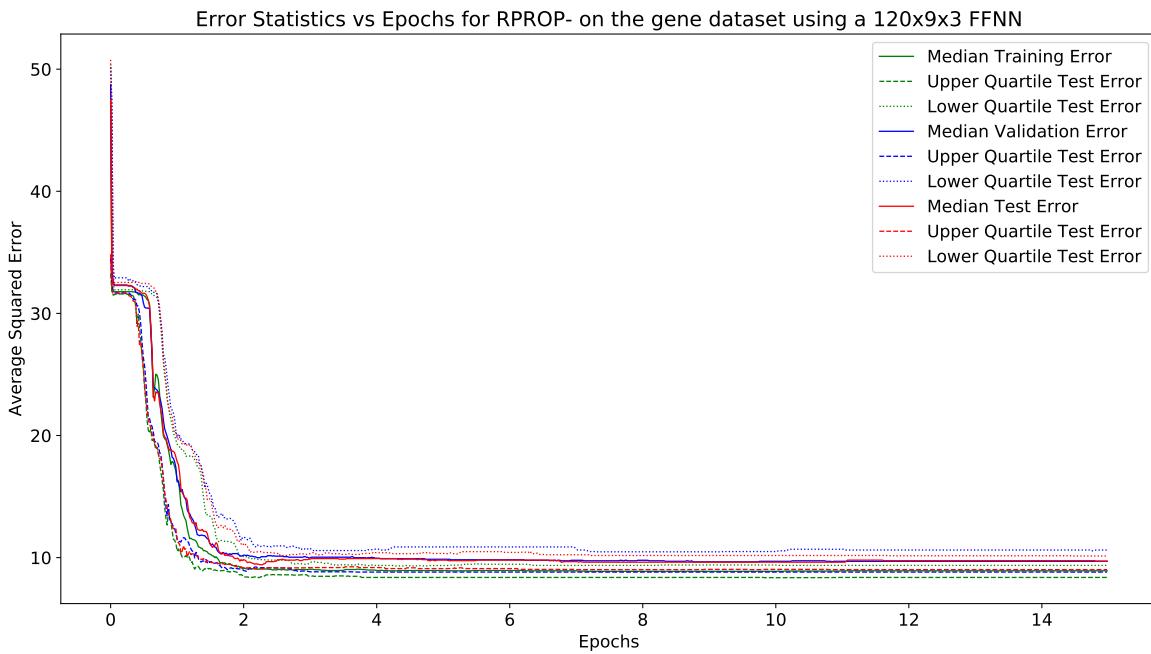
Fig. 353 shows the average and standard deviation of the test, training and validation errors. Fig. 354 shows the median and quartiles of the test, training and validation errors based on the best epoch. Finally, Fig. 355 shows the median and quartiles of the test, training and validation errors per mini-batch.



**Figure 353.** Graph of mean and standard deviation of errors vs epochs



**Figure 354.** Graph of test error vs epochs for the gradient based algorithms



**Figure 355.** Graph of test error vs epochs for the gradient based algorithms

#### 16.4.45 RPROP+

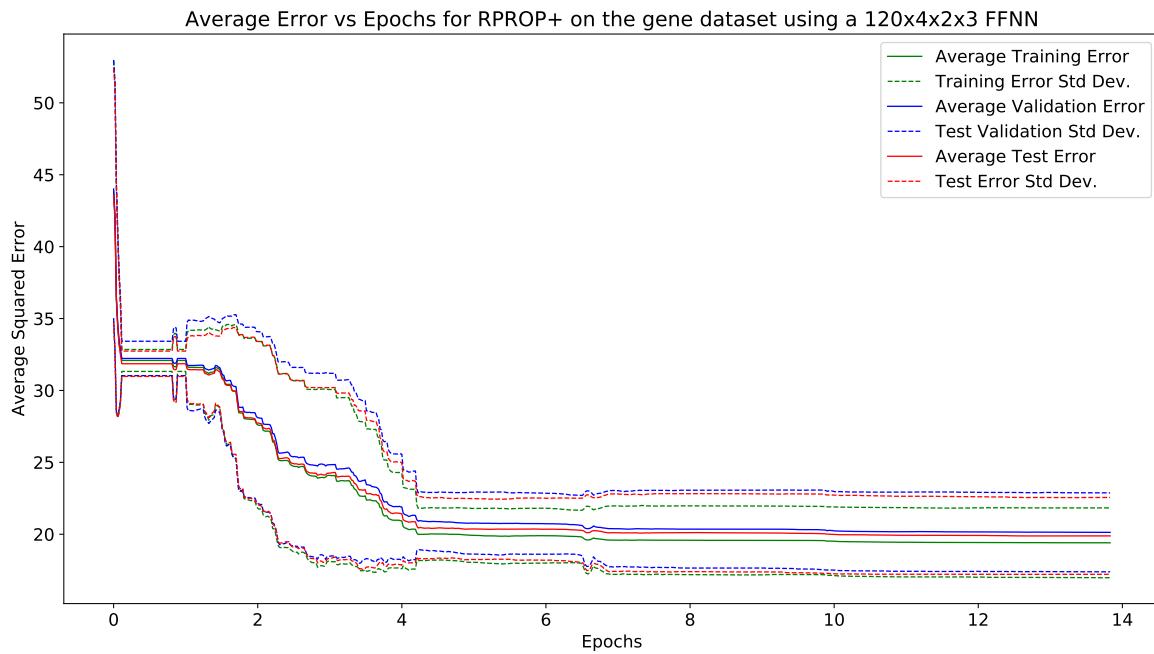
This section displays the results obtained using the RPROP+ algorithm.

#### 16.4.46 gene

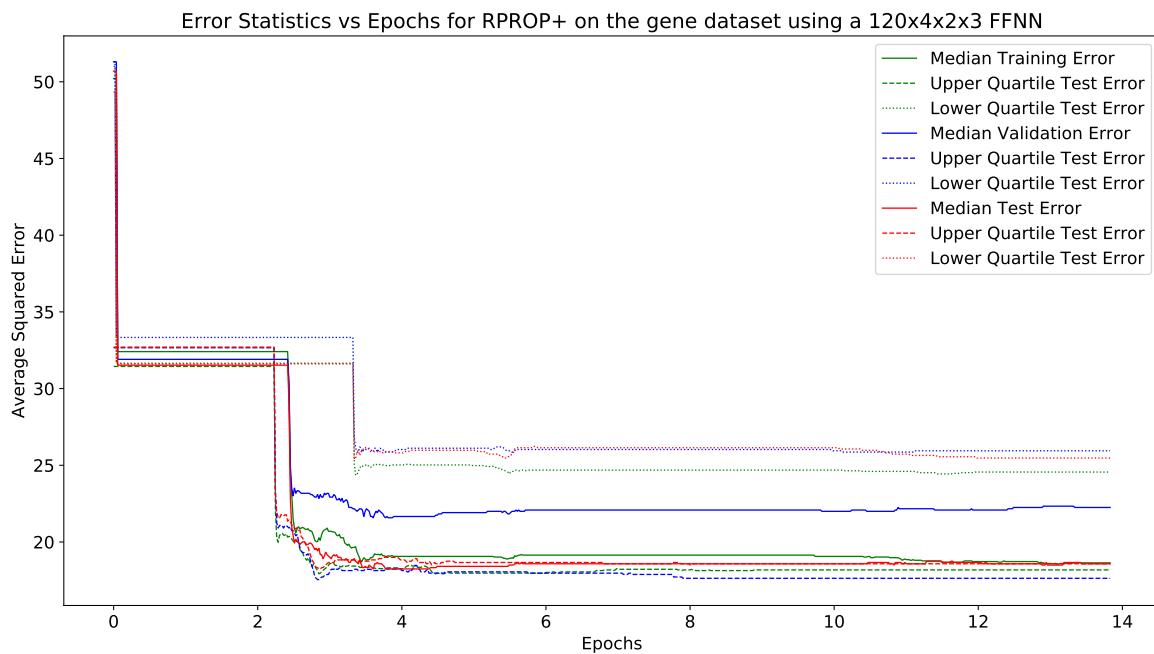
This section displays the results obtained using the gene algorithm.

##### 120 × 4 × 2 × 3 Architecture:

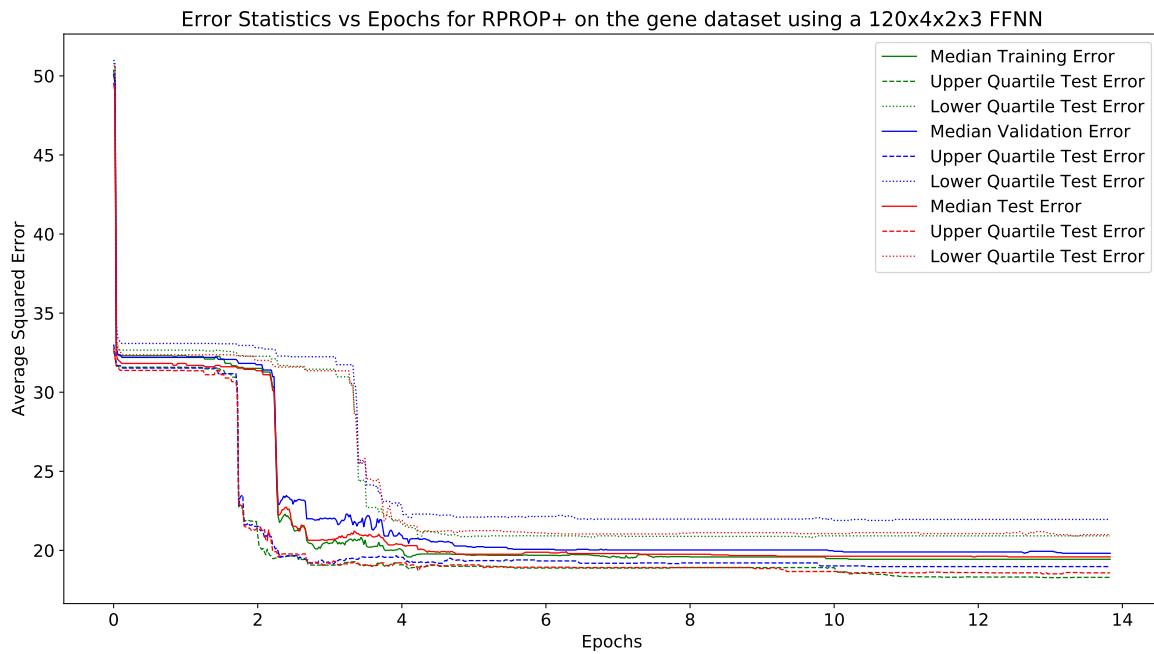
Fig. 356 shows the average and standard deviation of the test, training and validation errors. Fig. 357 shows the median and quartiles of the test, training and validation errors based on the best epoch. Finally, Fig. 358 shows the median and quartiles of the test, training and validation errors per mini-batch.



**Figure 356.** Graph of mean and standard deviation of errors vs epochs



**Figure 357.** Graph of test error vs epochs for the gradient based algorithms



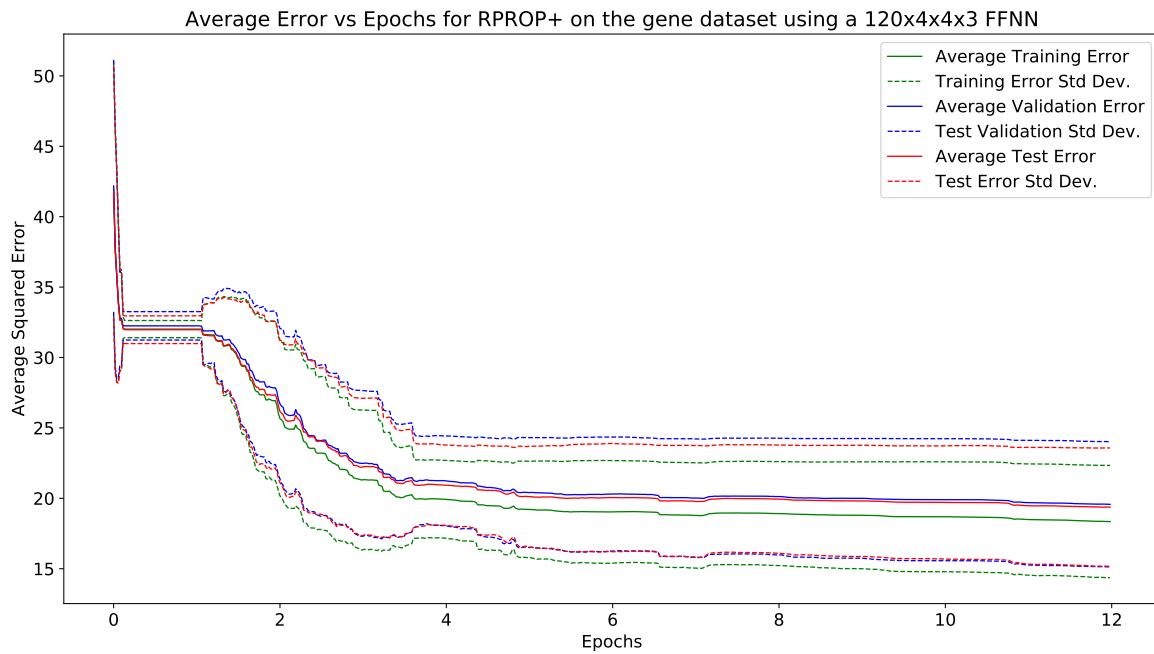
**Figure 358.** Graph of test error vs epochs for the gradient based algorithms

#### 16.4.47 gene

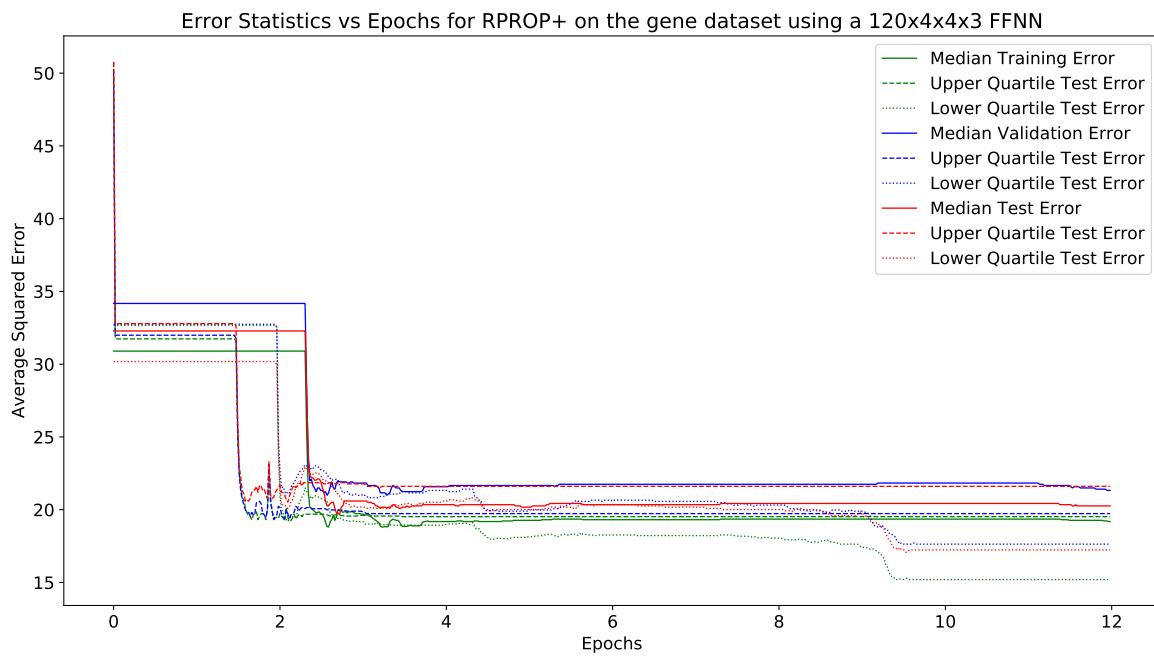
This section displays the results obtained using the gene algorithm.

##### 120 × 4 × 4 × 3 Architecture:

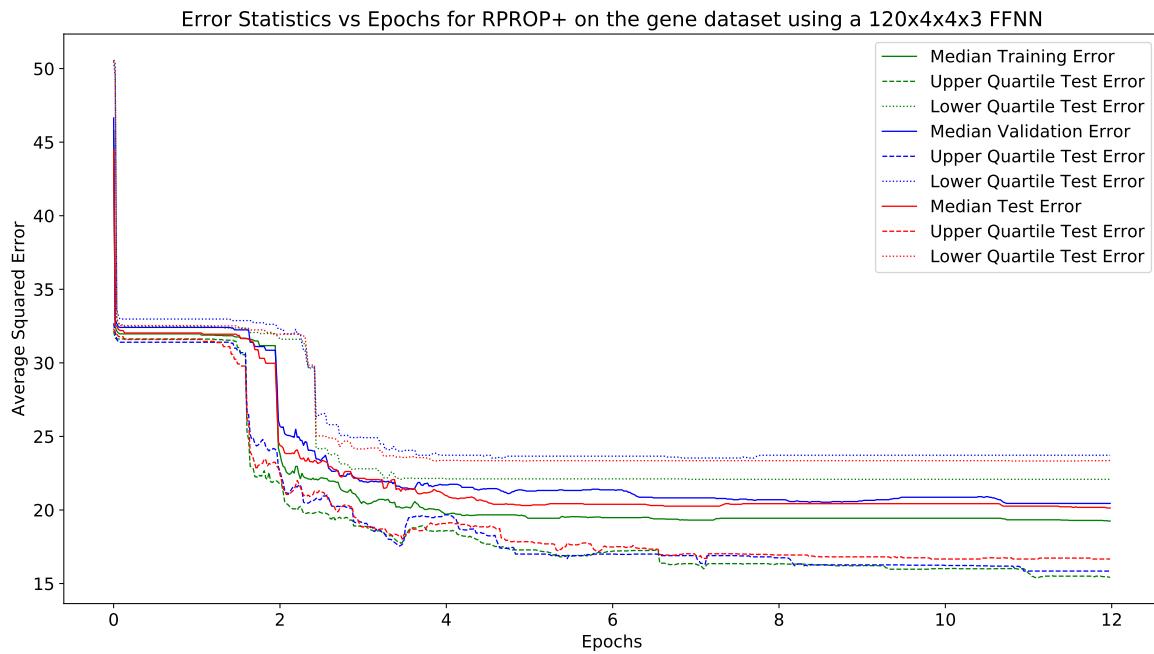
Fig. 359 shows the average and standard deviation of the test, training and validation errors. Fig. 360 shows the median and quartiles of the test, training and validation errors based on the best epoch. Finally, Fig. 361 shows the median and quartiles of the test, training and validation errors per mini-batch.



**Figure 359.** Graph of mean and standard deviation of errors vs epochs



**Figure 360.** Graph of test error vs epochs for the gradient based algorithms



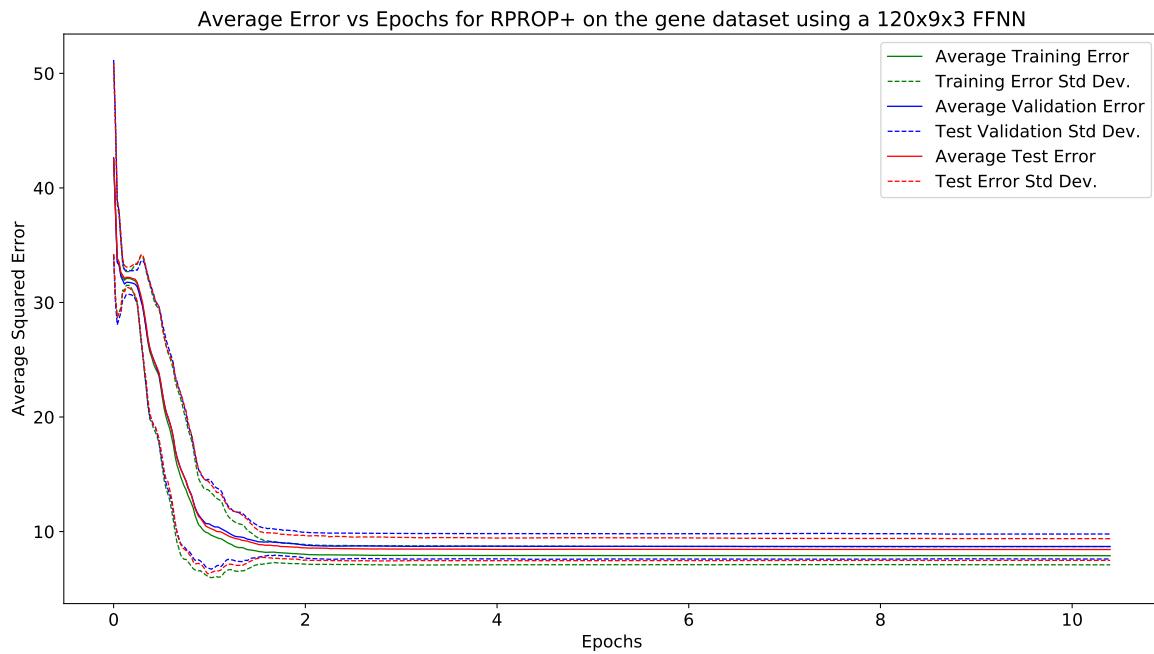
**Figure 361.** Graph of test error vs epochs for the gradient based algorithms

#### 16.4.48 gene

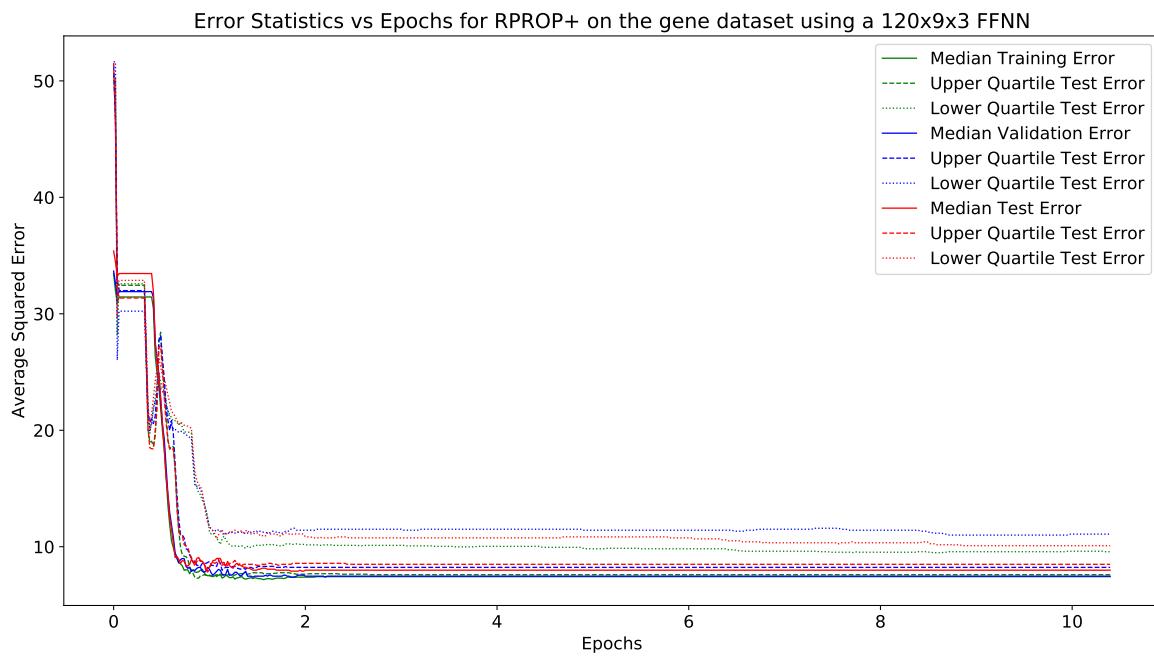
This section displays the results obtained using the gene algorithm.

##### 120 × 9 × 3 Architecture:

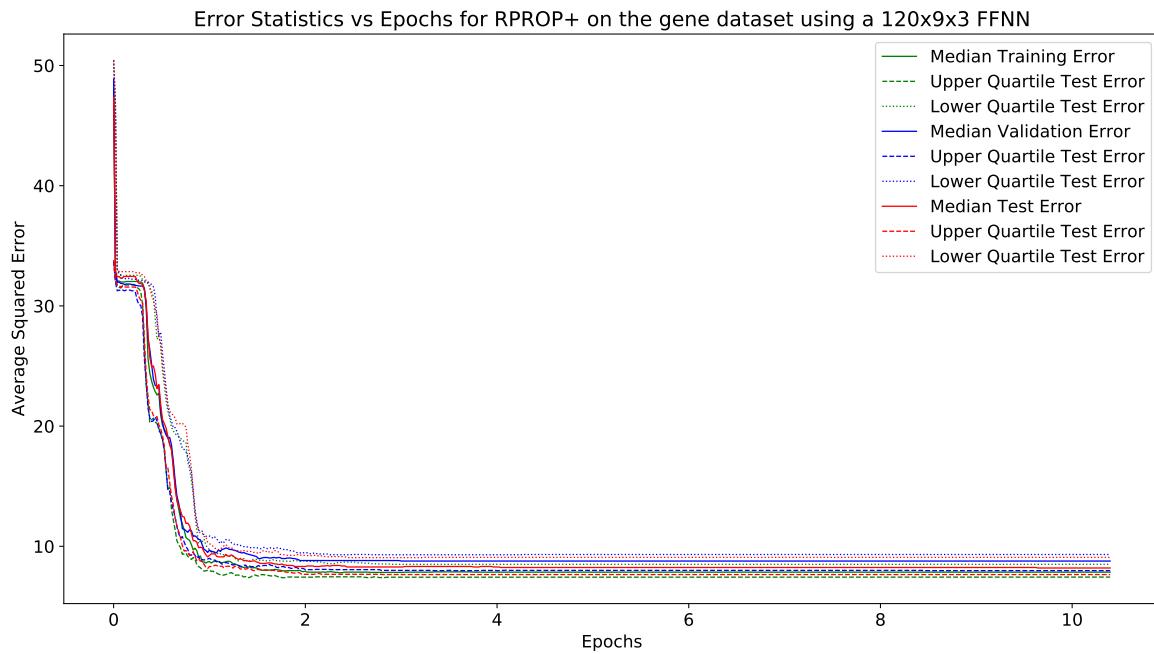
Fig. 362 shows the average and standard deviation of the test, training and validation errors. Fig. 363 shows the median and quartiles of the test, training and validation errors based on the best epoch. Finally, Fig. 364 shows the median and quartiles of the test, training and validation errors per mini-batch.



**Figure 362.** Graph of mean and standard deviation of errors vs epochs



**Figure 363.** Graph of test error vs epochs for the gradient based algorithms



**Figure 364.** Graph of test error vs epochs for the gradient based algorithms

## 16.5 horse

This section displays the results obtained in the horse dataset.

### 16.5.1 ADAGRAD

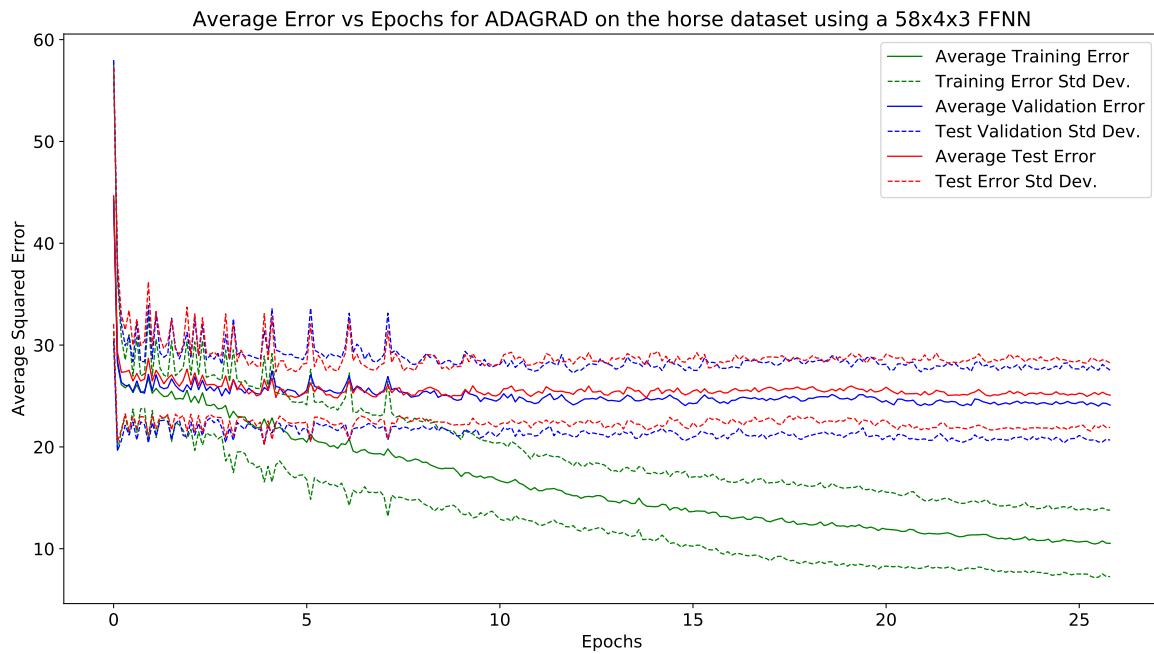
This section displays the results obtained using the ADAGRAD algorithm.

### 16.5.2 horse

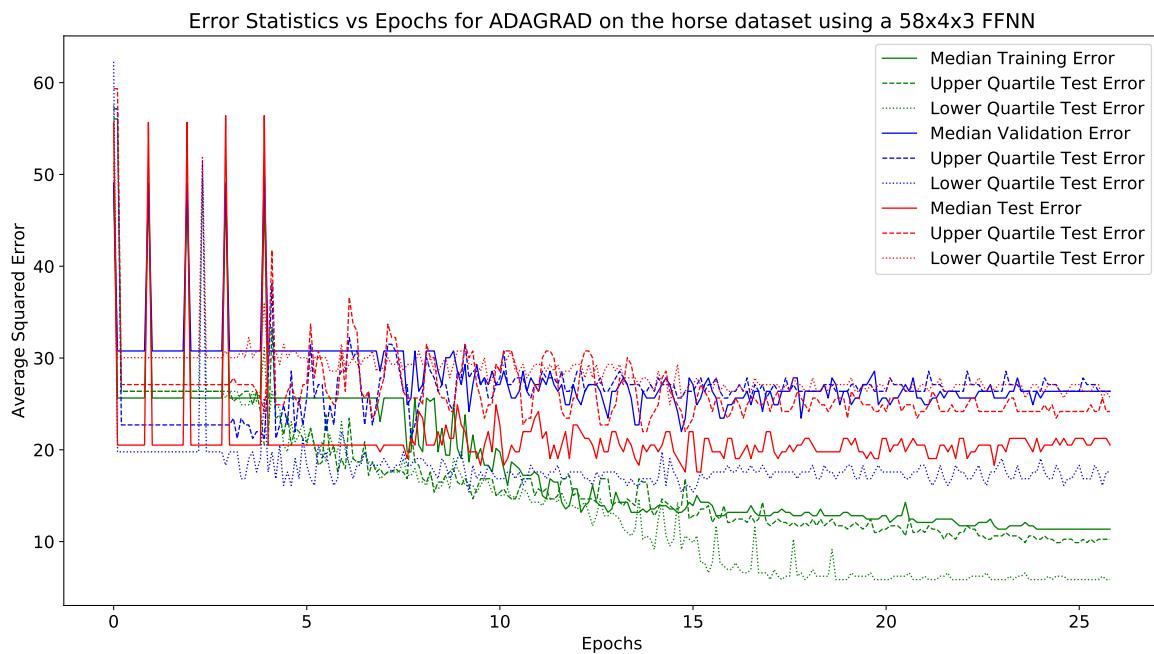
This section displays the results obtained using the horse algorithm.

#### 58 × 4 × 3 Architecture:

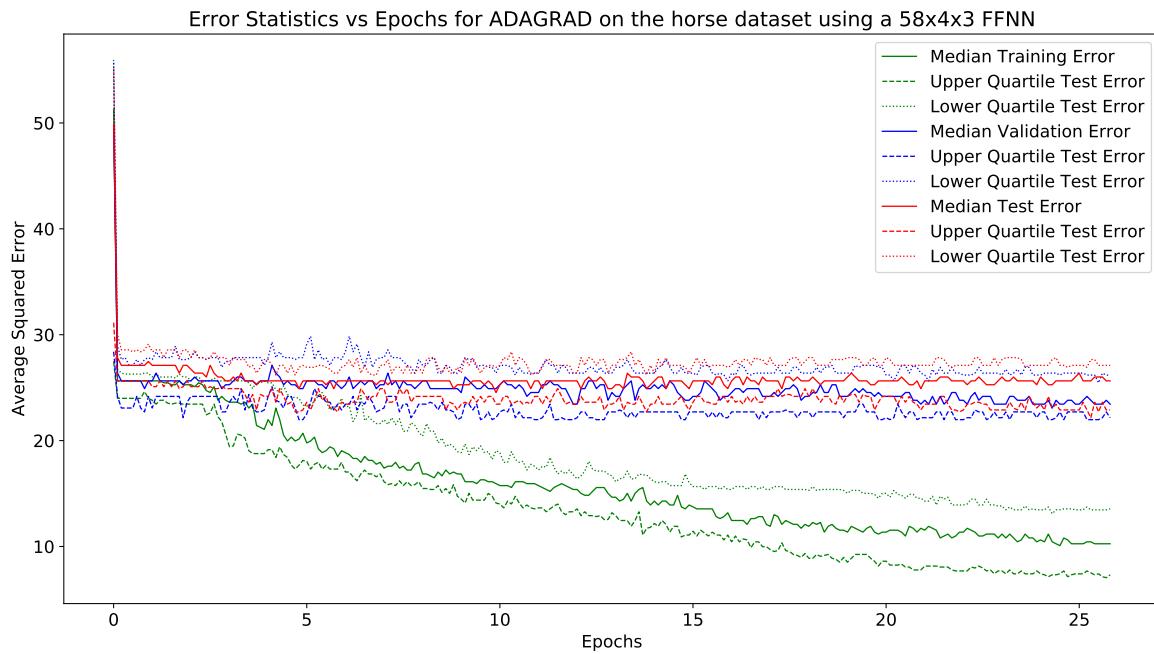
Fig. 365 shows the average and standard deviation of the test, training and validation errors. Fig. 366 shows the median and quartiles of the test, training and validation errors based on the best epoch. Finally, Fig. 367 shows the median and quartiles of the test, training and validation errors per mini-batch.



**Figure 365.** Graph of mean and standard deviation of errors vs epochs



**Figure 366.** Graph of test error vs epochs for the gradient based algorithms



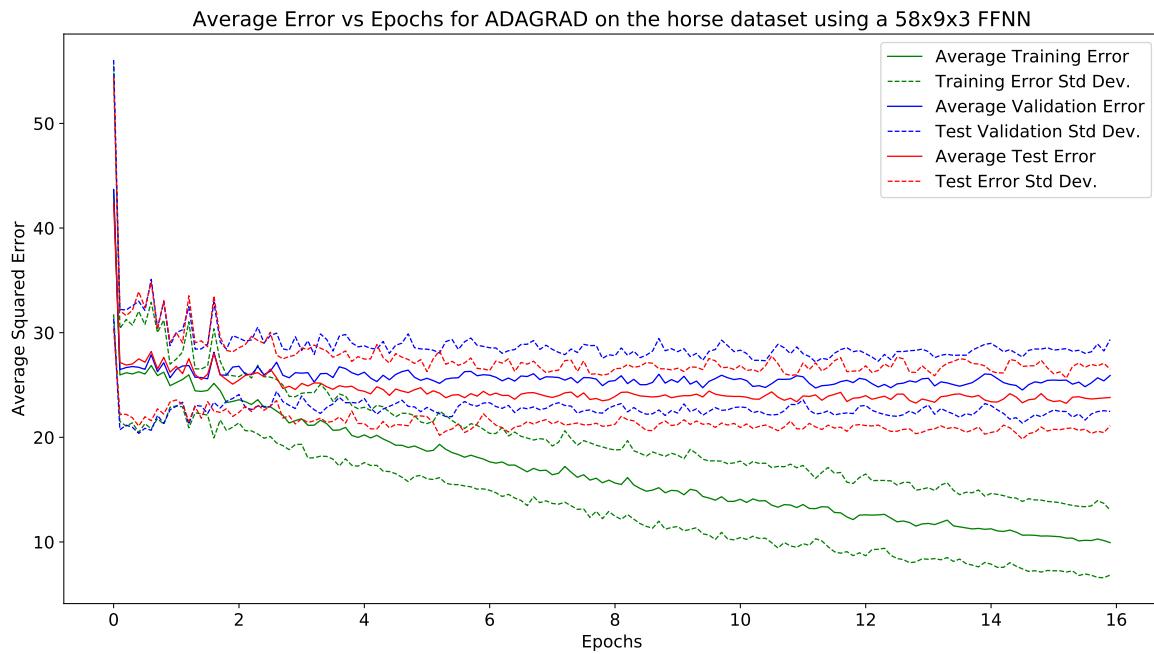
**Figure 367.** Graph of test error vs epochs for the gradient based algorithms

### 16.5.3 horse

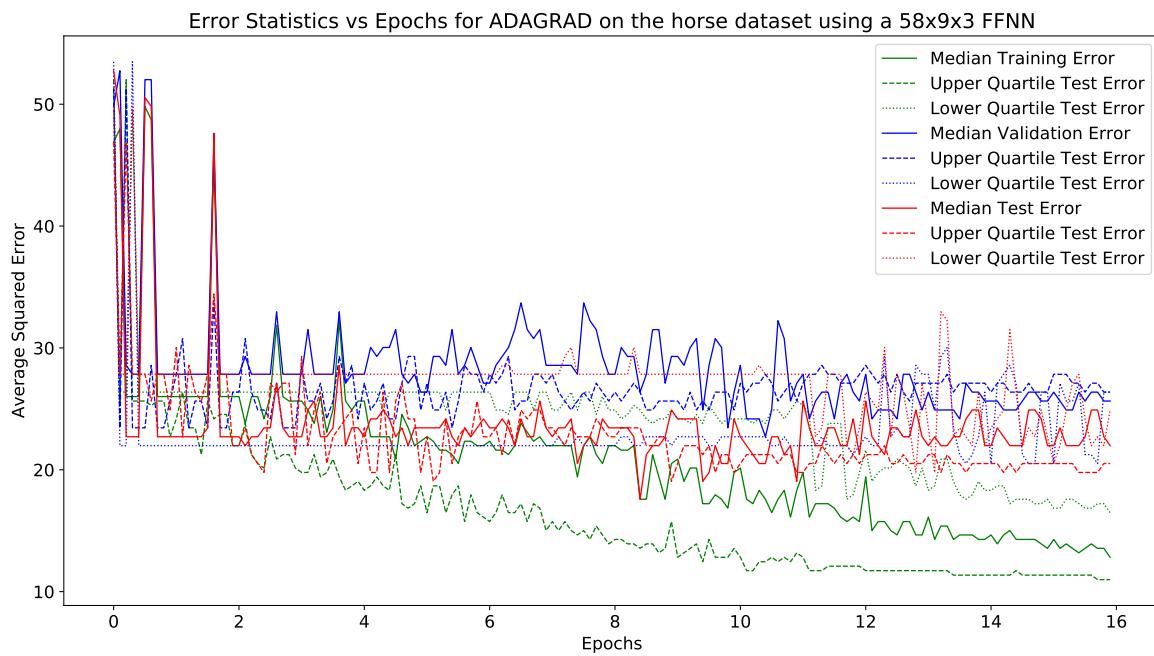
This section displays the results obtained using the horse algorithm.

#### 58 × 9 × 3 Architecture:

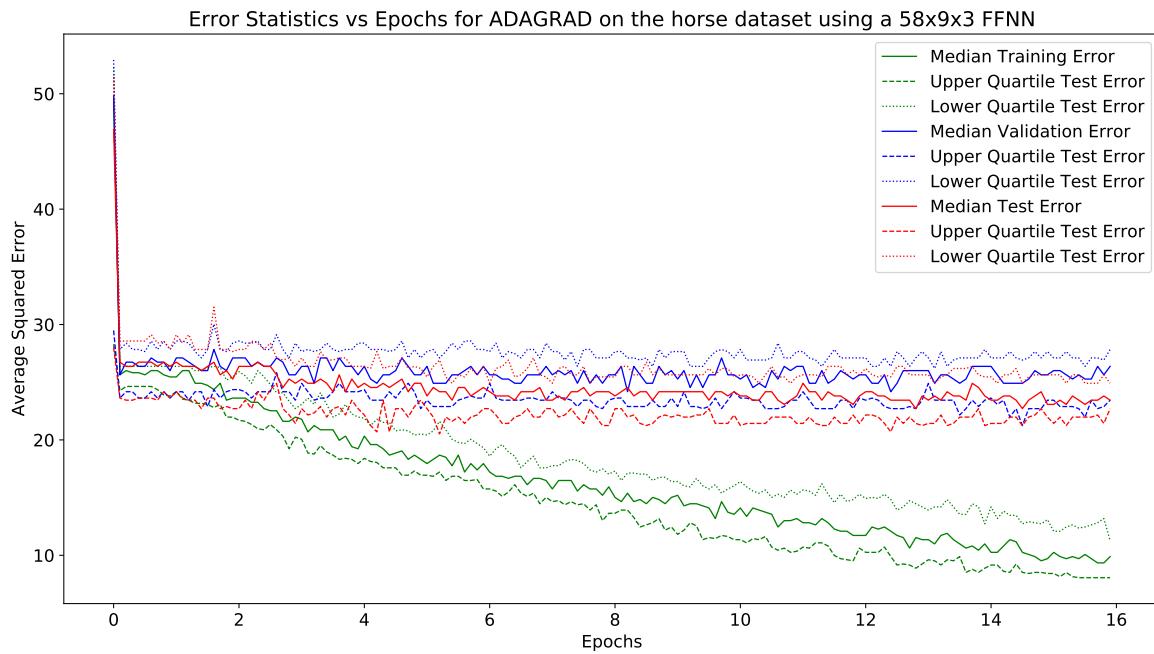
Fig. 368 shows the average and standard deviation of the test, training and validation errors. Fig. 369 shows the median and quartiles of the test, training and validation errors based on the best epoch. Finally, Fig. 370 shows the median and quartiles of the test, training and validation errors per mini-batch.



**Figure 368.** Graph of mean and standard deviation of errors vs epochs



**Figure 369.** Graph of test error vs epochs for the gradient based algorithms



**Figure 370.** Graph of test error vs epochs for the gradient based algorithms

#### 16.5.4 Backprop with Momentum

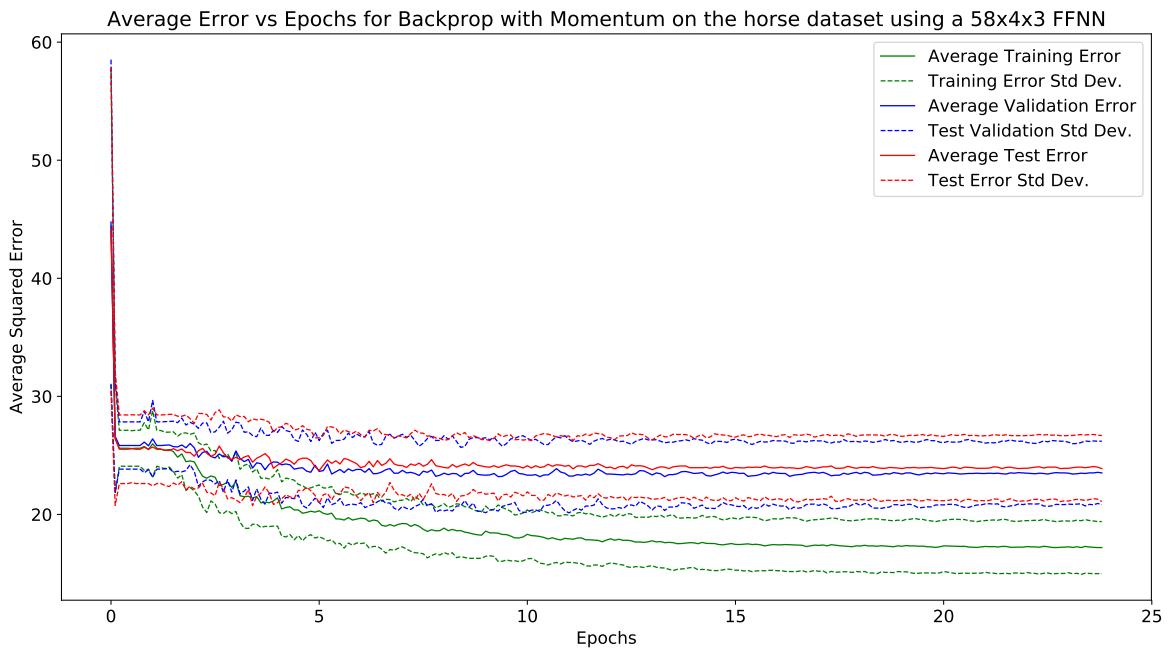
This section displays the results obtained using the Backprop with Momentum algorithm.

#### 16.5.5 horse

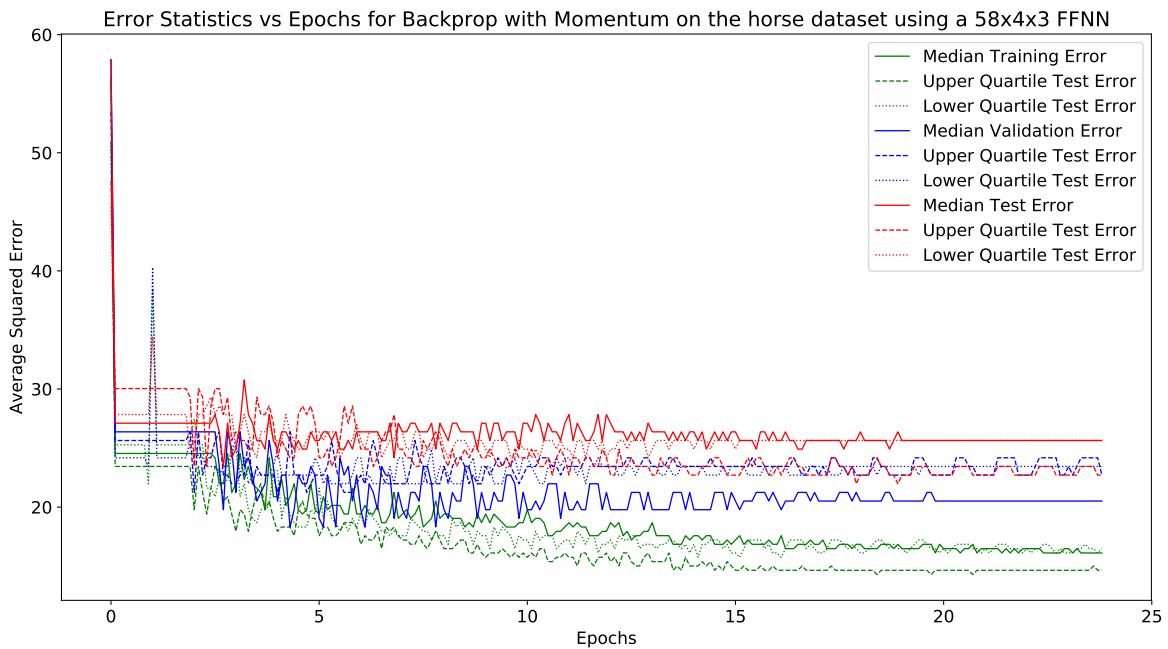
This section displays the results obtained using the horse algorithm.

##### 58 × 4 × 3 Architecture:

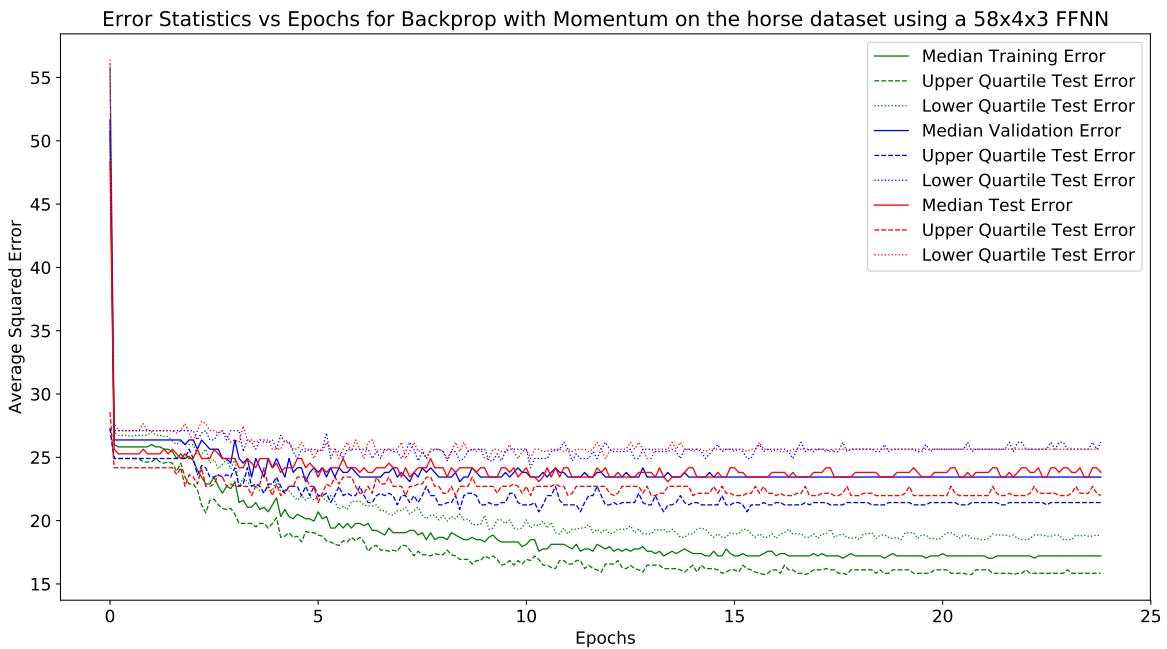
Fig. 371 shows the average and standard deviation of the test, training and validation errors. Fig. 372 shows the median and quartiles of the test, training and validation errors based on the best epoch. Finally, Fig. 373 shows the median and quartiles of the test, training and validation errors per mini-batch.



**Figure 371.** Graph of mean and standard deviation of errors vs epochs



**Figure 372.** Graph of test error vs epochs for the gradient based algorithms



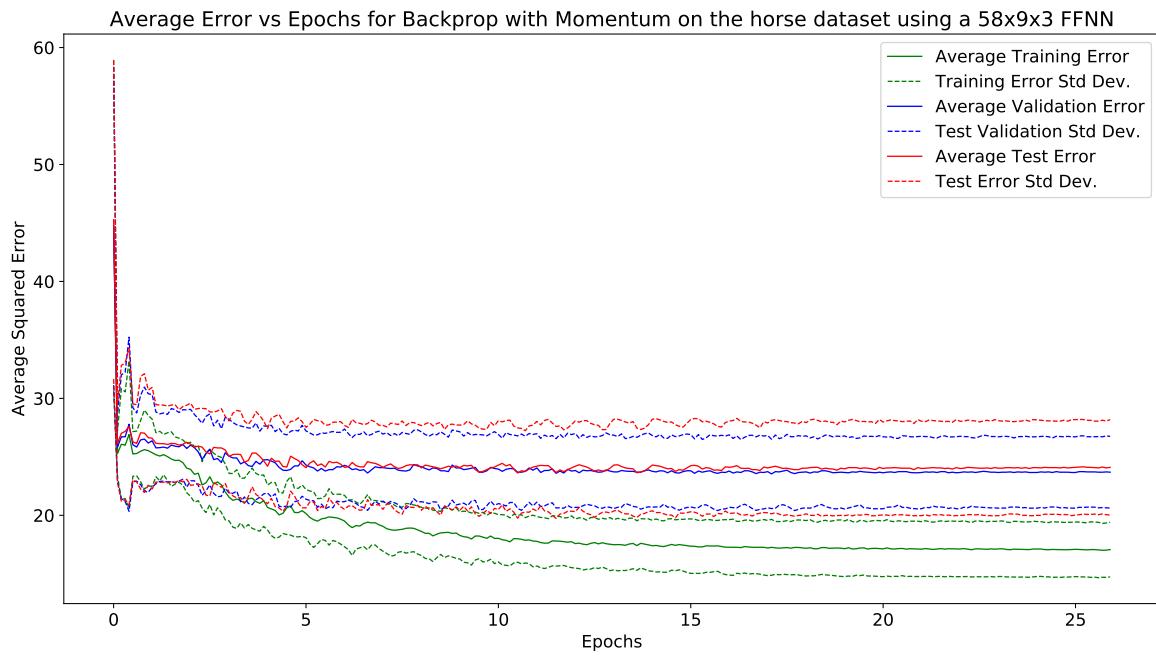
**Figure 373.** Graph of test error vs epochs for the gradient based algorithms

### 16.5.6 horse

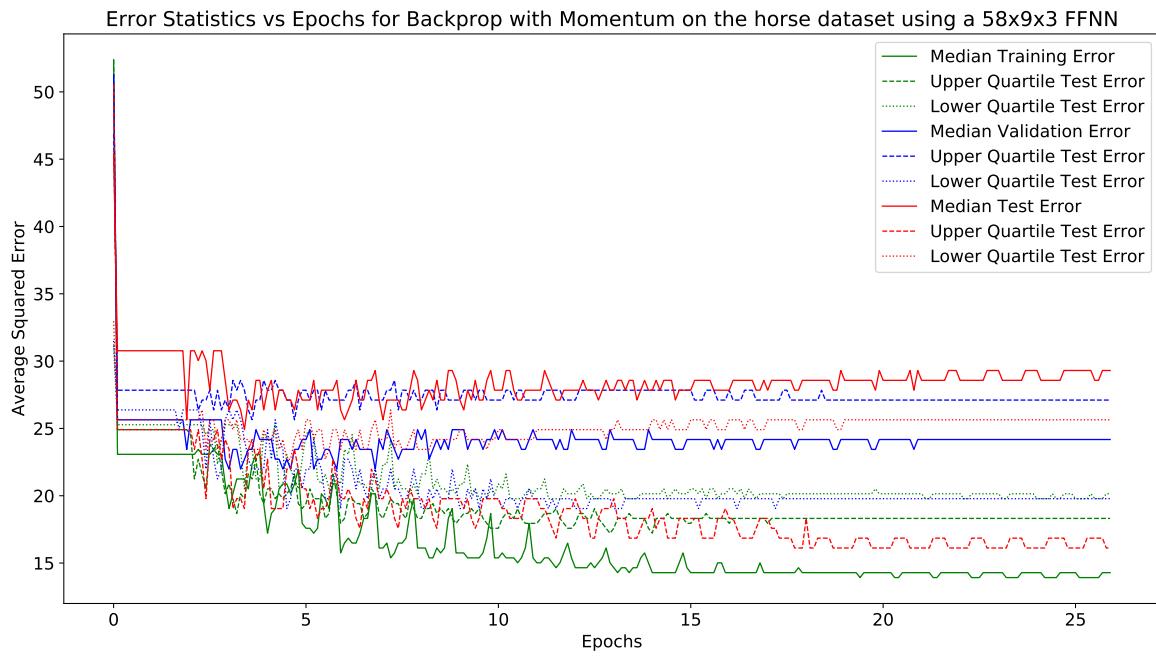
This section displays the results obtained using the horse algorithm.

#### 58 × 9 × 3 Architecture:

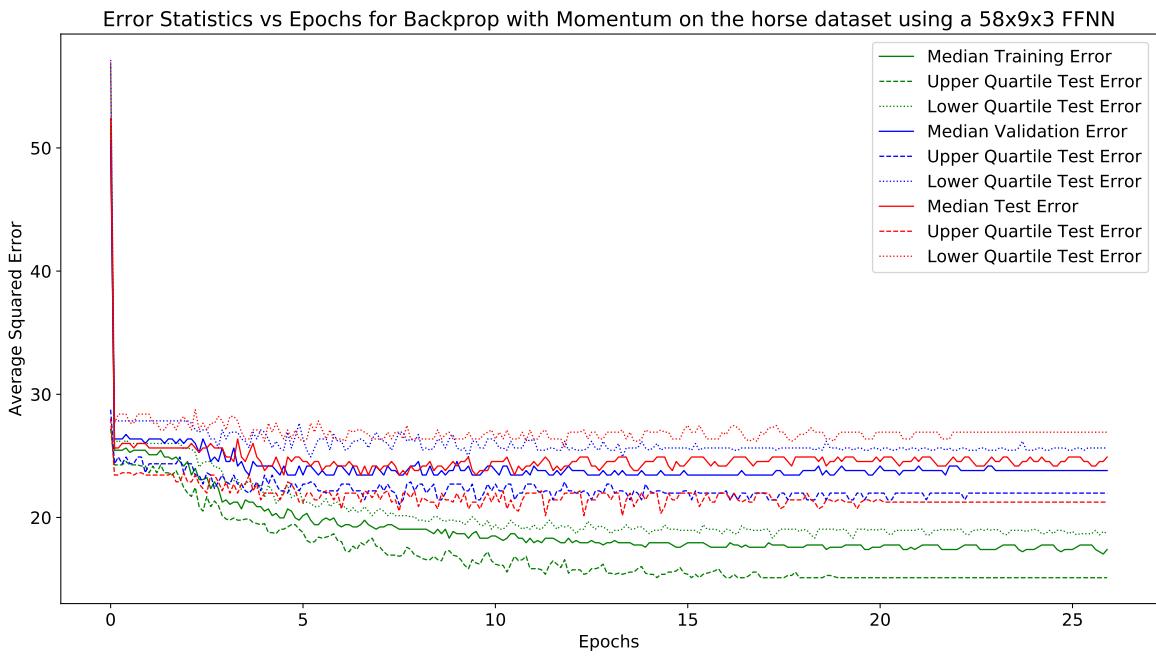
Fig. 374 shows the average and standard deviation of the test, training and validation errors. Fig. 375 shows the median and quartiles of the test, training and validation errors based on the best epoch. Finally, Fig. 376 shows the median and quartiles of the test, training and validation errors per mini-batch.



**Figure 374.** Graph of mean and standard deviation of errors vs epochs



**Figure 375.** Graph of test error vs epochs for the gradient based algorithms



**Figure 376.** Graph of test error vs epochs for the gradient based algorithms

### 16.5.7 back-propagation

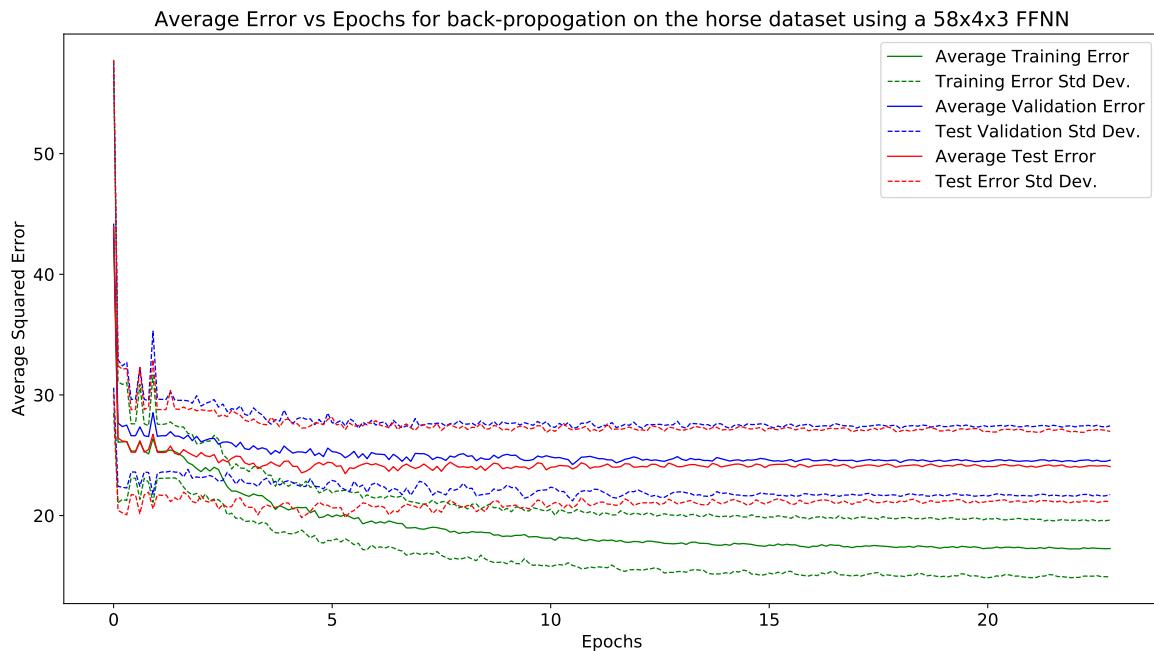
This section displays the results obtained using the back-propagation algorithm.

### 16.5.8 horse

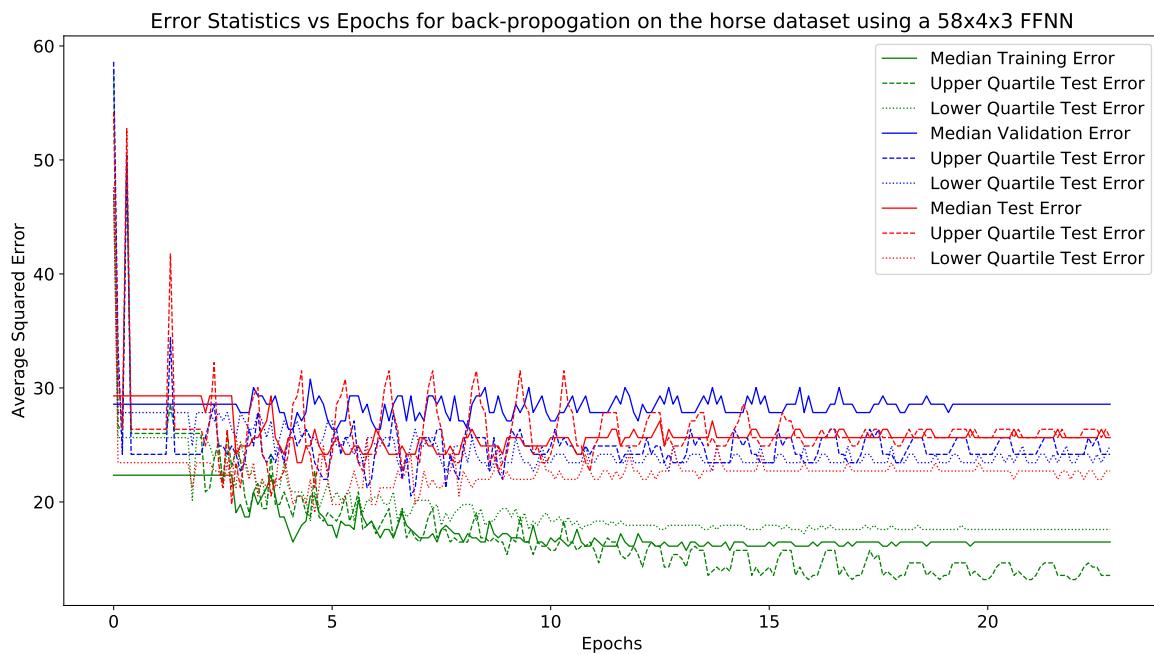
This section displays the results obtained using the horse algorithm.

#### 58 × 4 × 3 Architecture:

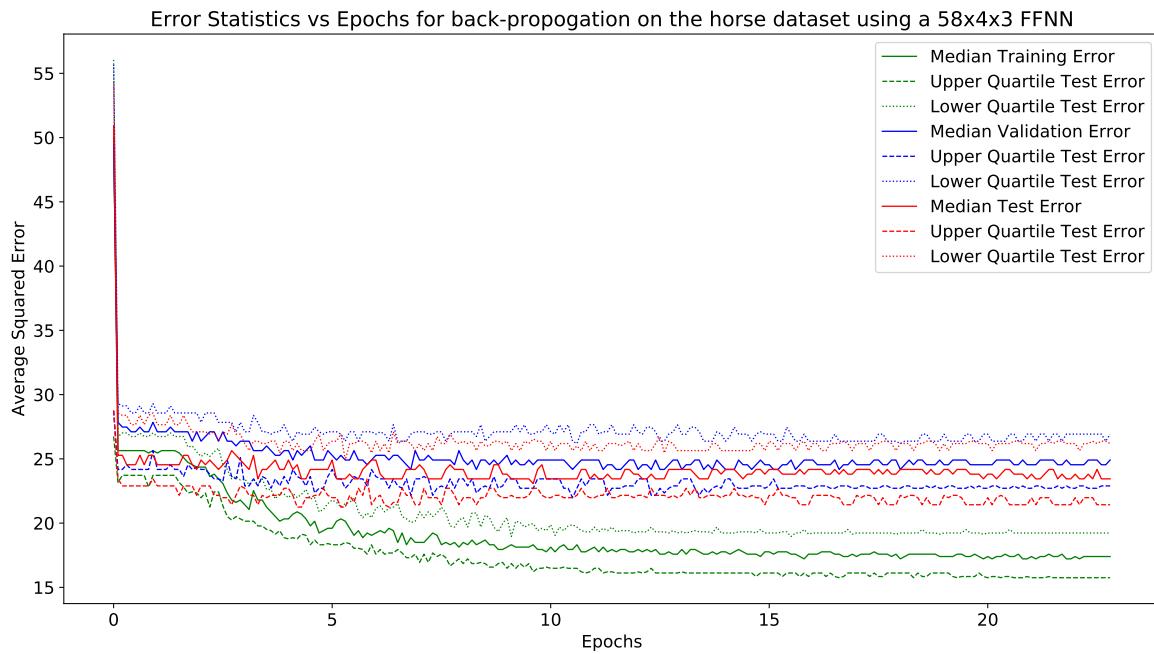
Fig. 377 shows the average and standard deviation of the test, training and validation errors. Fig. 378 shows the median and quartiles of the test, training and validation errors based on the best epoch. Finally, Fig. 379 shows the median and quartiles of the test, training and validation errors per mini-batch.



**Figure 377.** Graph of mean and standard deviation of errors vs epochs



**Figure 378.** Graph of test error vs epochs for the gradient based algorithms



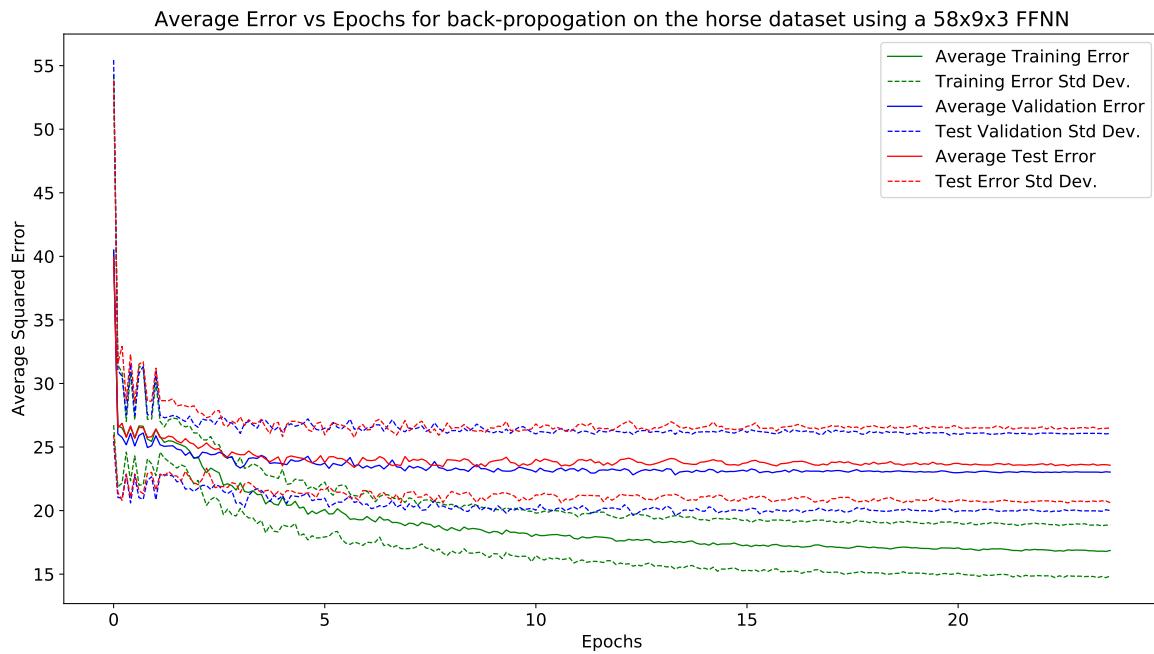
**Figure 379.** Graph of test error vs epochs for the gradient based algorithms

### 16.5.9 horse

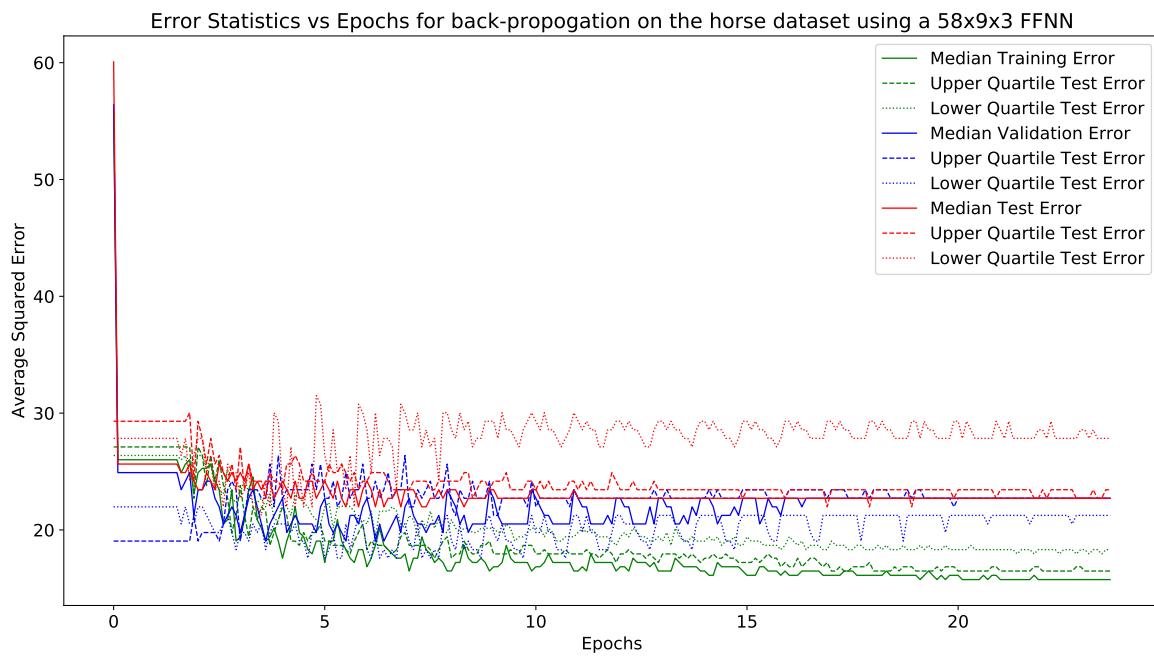
This section displays the results obtained using the horse algorithm.

#### 58 × 9 × 3 Architecture:

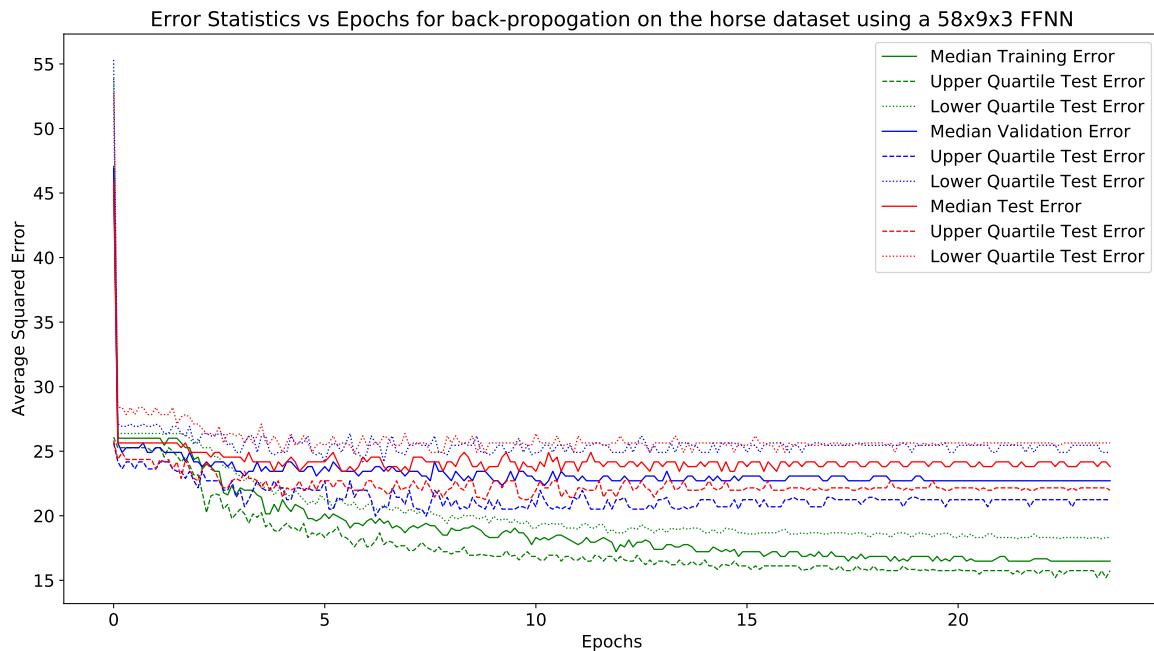
Fig. 380 shows the average and standard deviation of the test, training and validation errors. Fig. 381 shows the median and quartiles of the test, training and validation errors based on the best epoch. Finally, Fig. 382 shows the median and quartiles of the test, training and validation errors per mini-batch.



**Figure 380.** Graph of mean and standard deviation of errors vs epochs



**Figure 381.** Graph of test error vs epochs for the gradient based algorithms



**Figure 382.** Graph of test error vs epochs for the gradient based algorithms

### 16.5.10 GA1

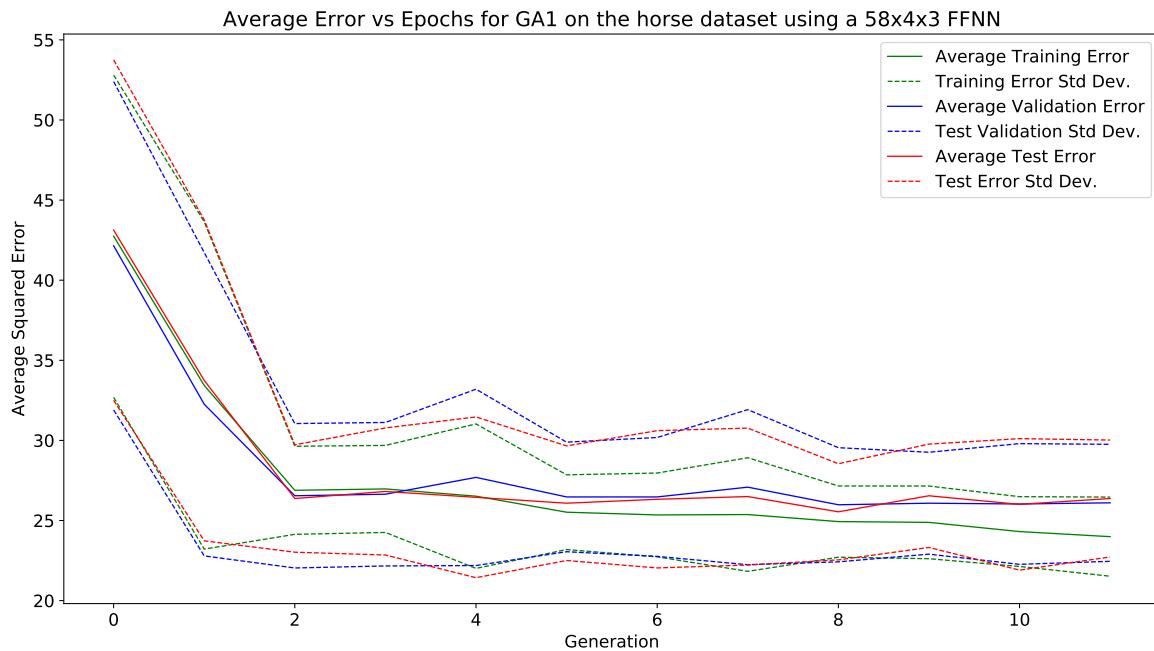
This section displays the results obtained using the GA1 algorithm.

### 16.5.11 horse

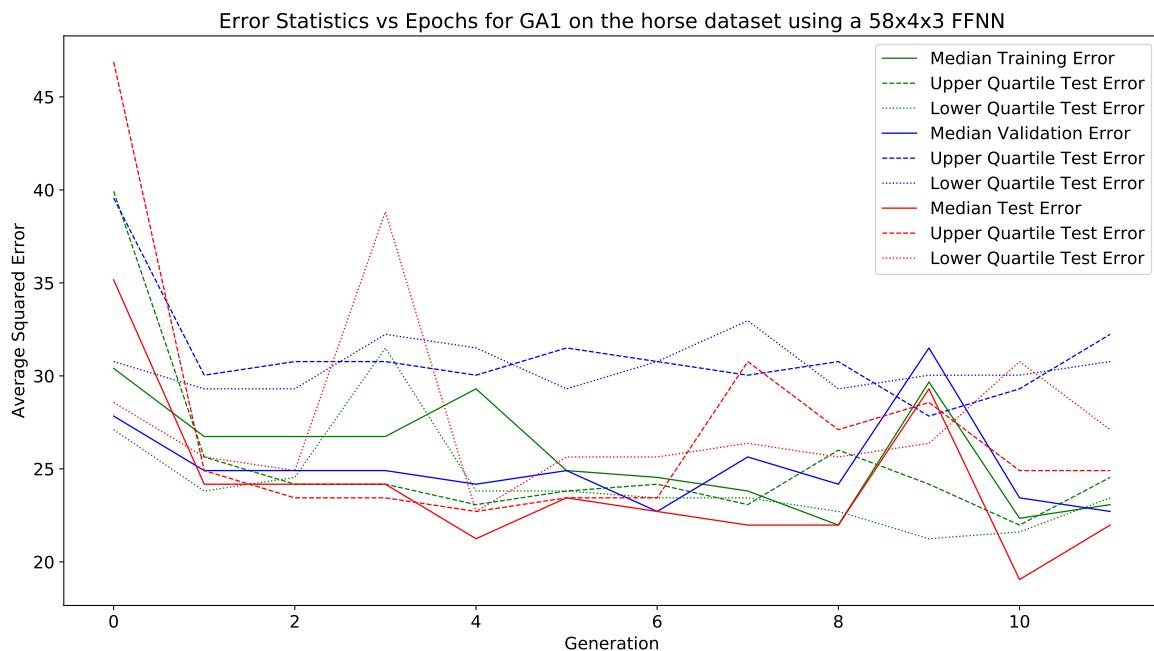
This section displays the results obtained using the horse algorithm.

#### 58 × 4 × 3 Architecture:

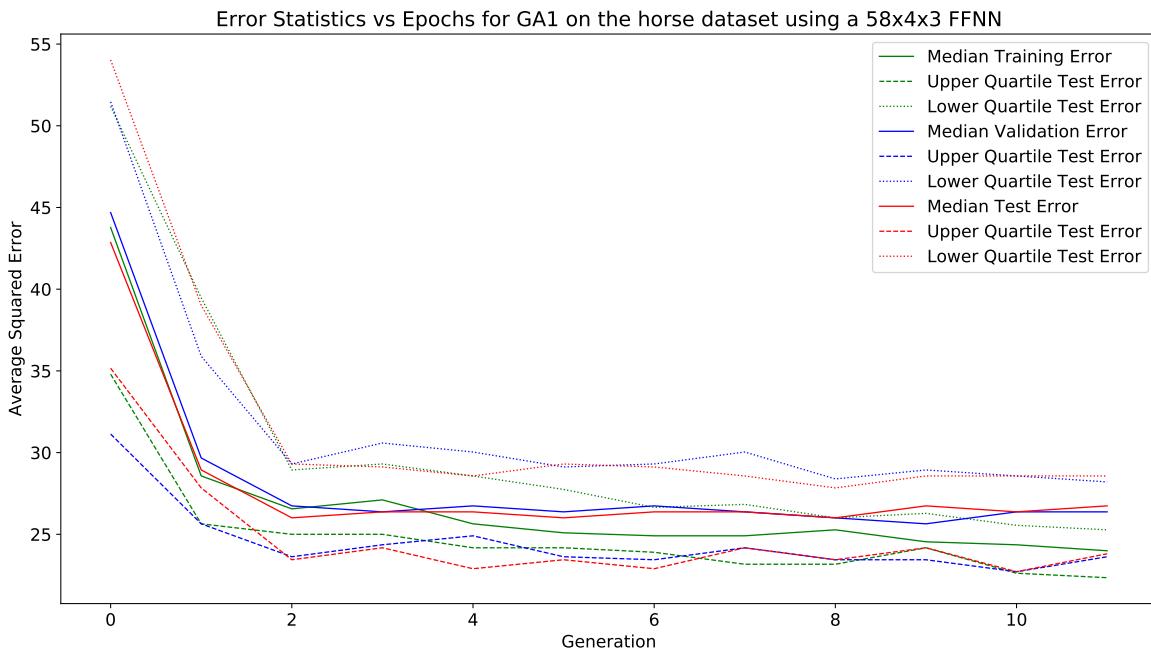
Fig. 383 shows the average and standard deviation of the test, training and validation errors. Fig. 384 shows the median and quartiles of the test, training and validation errors based on the best epoch. Finally, Fig. 385 shows the median and quartiles of the test, training and validation errors per mini-batch.



**Figure 383.** Graph of mean and standard deviation of errors vs epochs



**Figure 384.** Graph of test error vs epochs for the gradient based algorithms



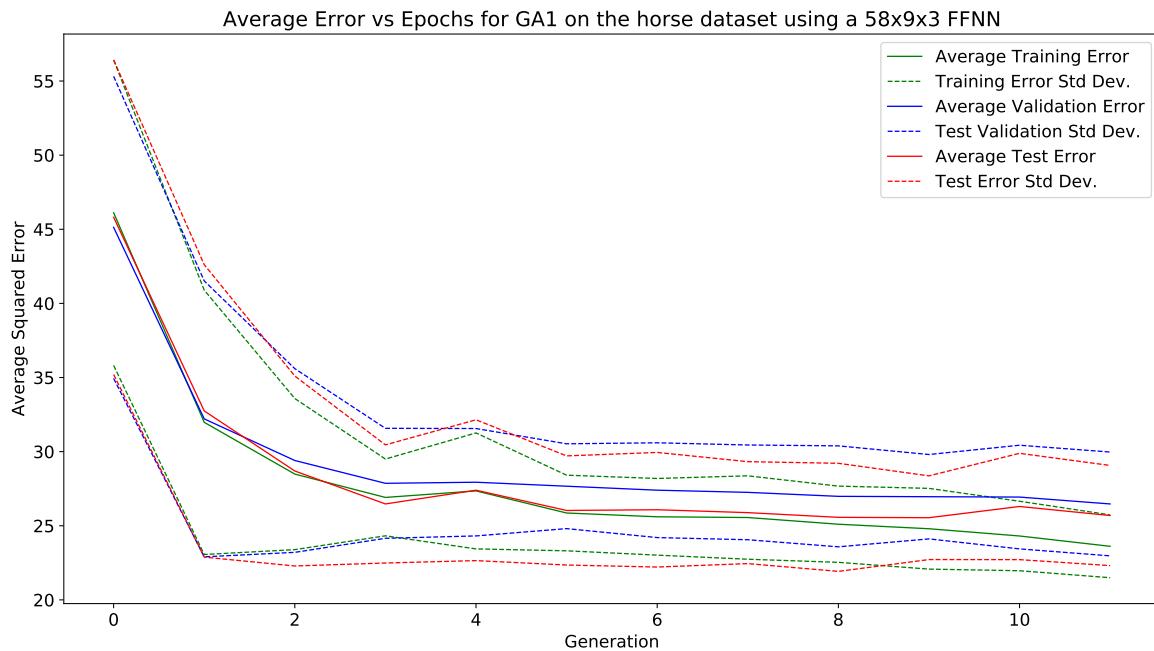
**Figure 385.** Graph of test error vs epochs for the gradient based algorithms

### 16.5.12 horse

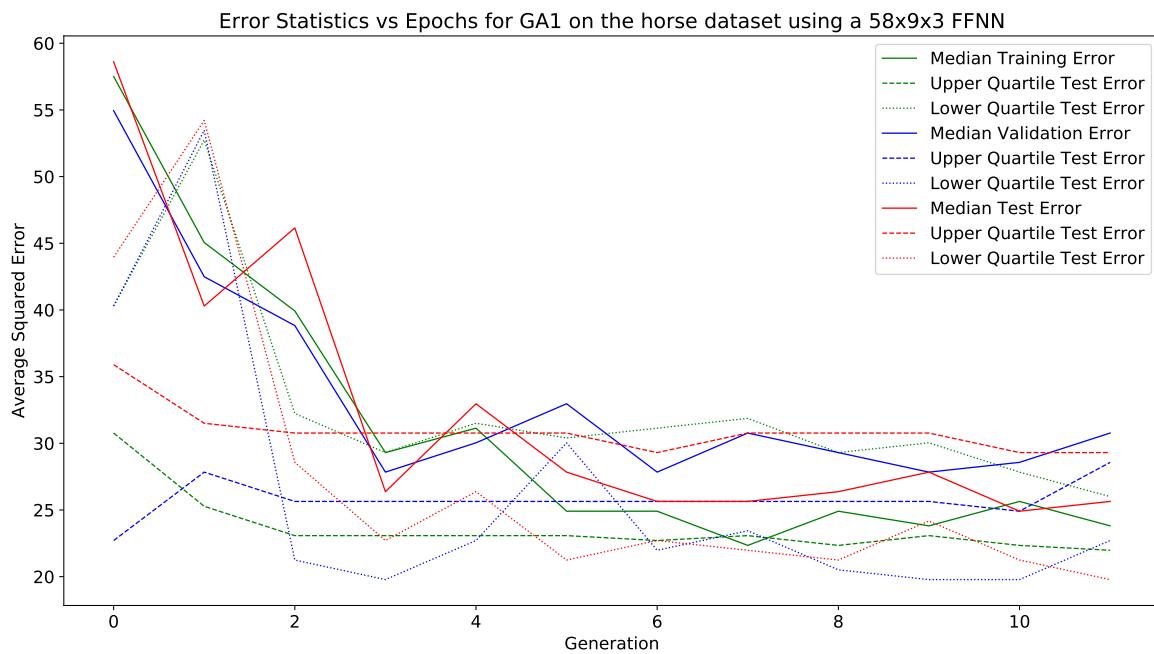
This section displays the results obtained using the horse algorithm.

#### 58 × 9 × 3 Architecture:

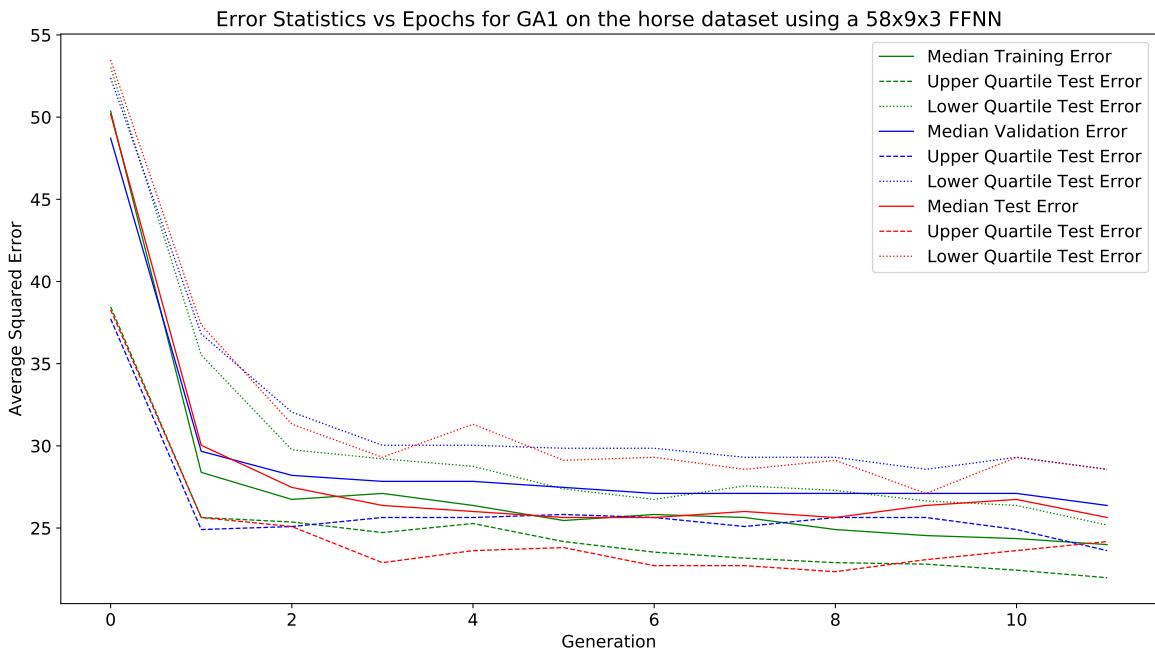
Fig. 386 shows the average and standard deviation of the test, training and validation errors. Fig. 387 shows the median and quartiles of the test, training and validation errors based on the best epoch. Finally, Fig. 388 shows the median and quartiles of the test, training and validation errors per mini-batch.



**Figure 386.** Graph of mean and standard deviation of errors vs epochs



**Figure 387.** Graph of test error vs epochs for the gradient based algorithms



**Figure 388.** Graph of test error vs epochs for the gradient based algorithms

### 16.5.13 GA2

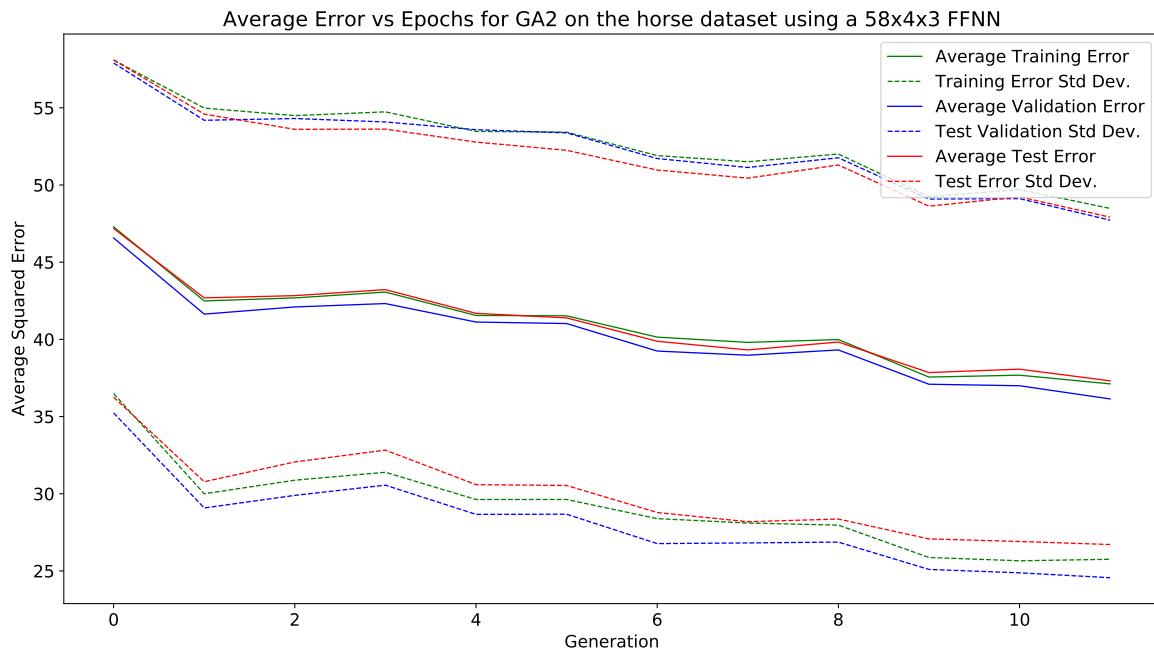
This section displays the results obtained using the GA2 algorithm.

### 16.5.14 horse

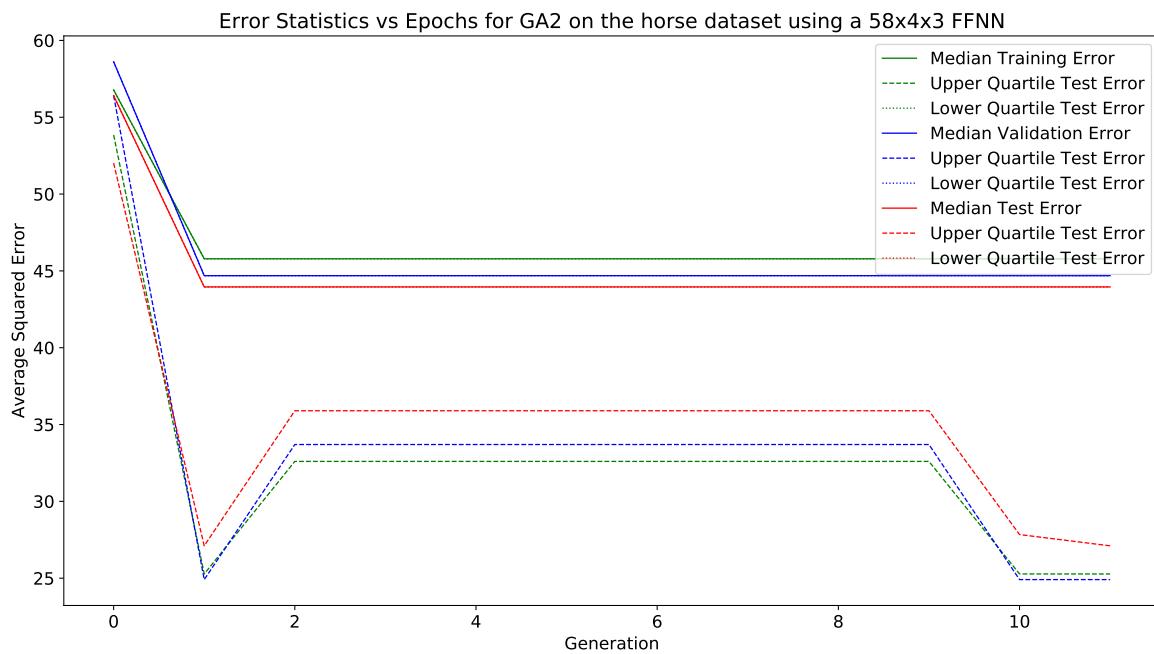
This section displays the results obtained using the horse algorithm.

#### 58 × 4 × 3 Architecture:

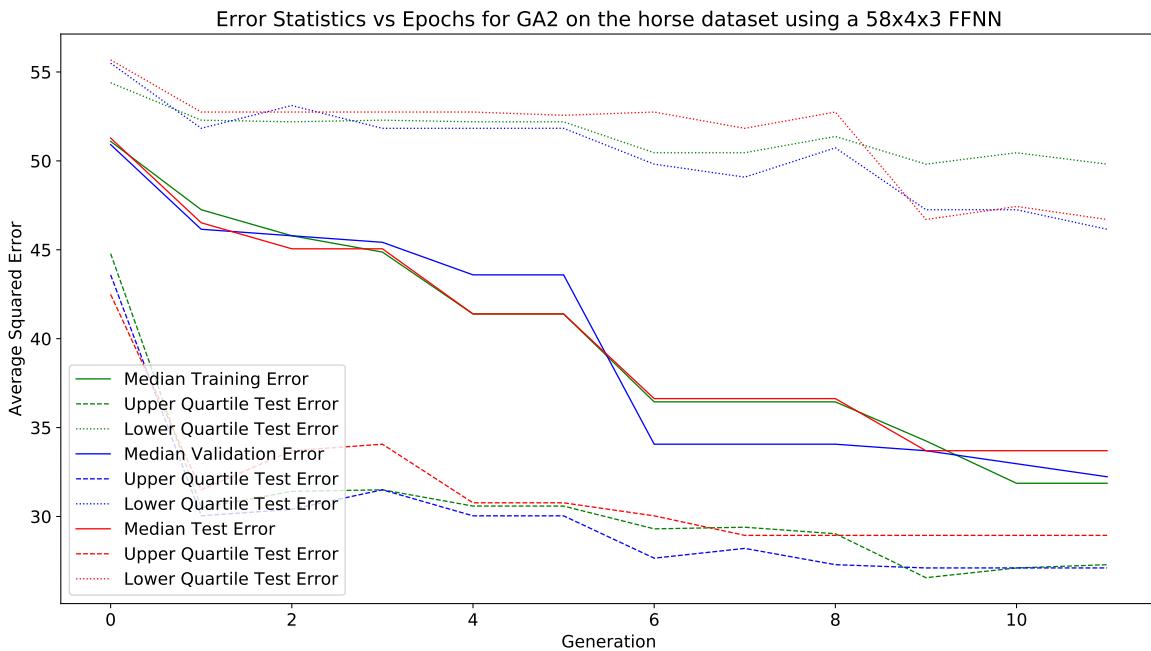
Fig. 389 shows the average and standard deviation of the test, training and validation errors. Fig. 390 shows the median and quartiles of the test, training and validation errors based on the best epoch. Finally, Fig. 391 shows the median and quartiles of the test, training and validation errors per mini-batch.



**Figure 389.** Graph of mean and standard deviation of errors vs epochs



**Figure 390.** Graph of test error vs epochs for the gradient based algorithms



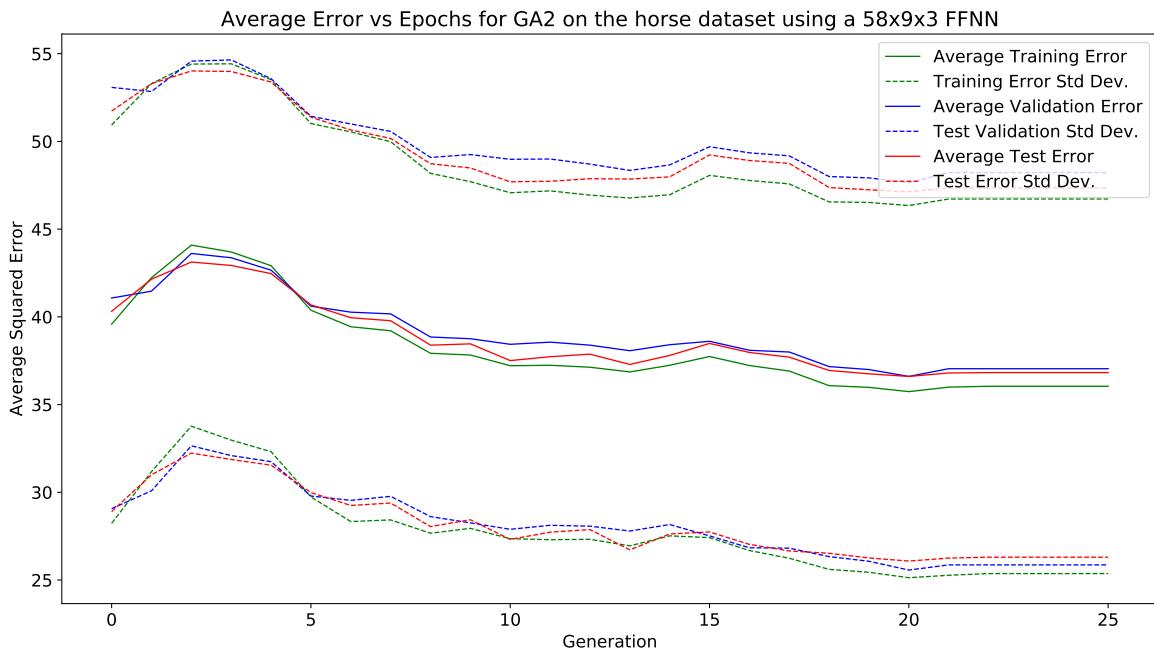
**Figure 391.** Graph of test error vs epochs for the gradient based algorithms

### 16.5.15 horse

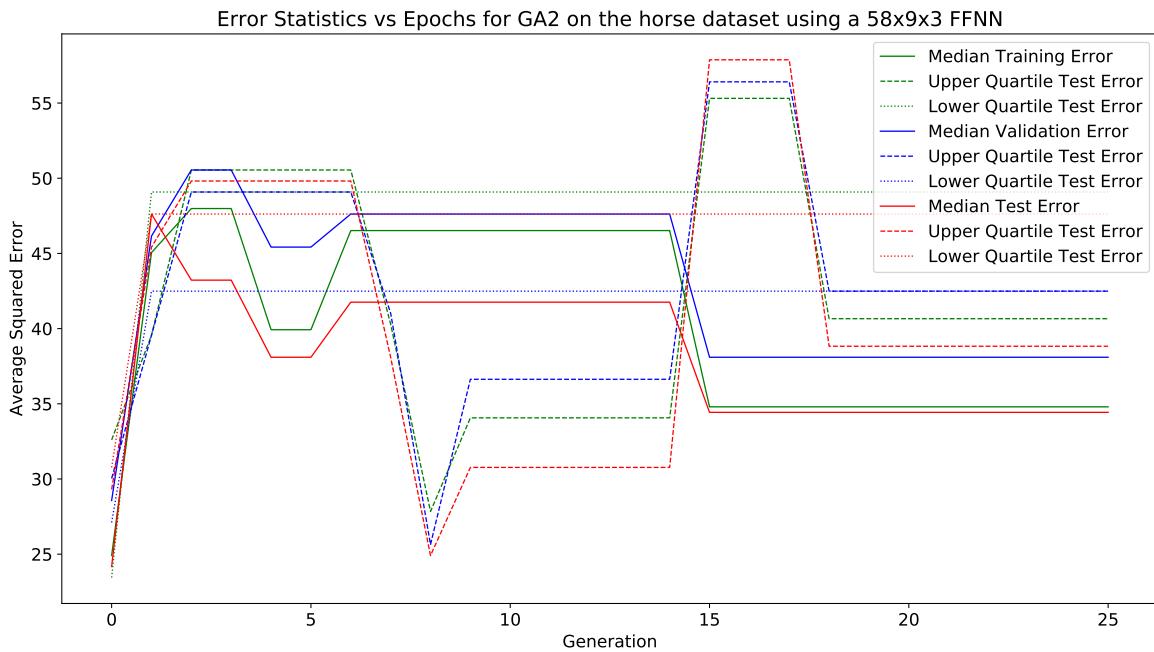
This section displays the results obtained using the horse algorithm.

#### 58 × 9 × 3 Architecture:

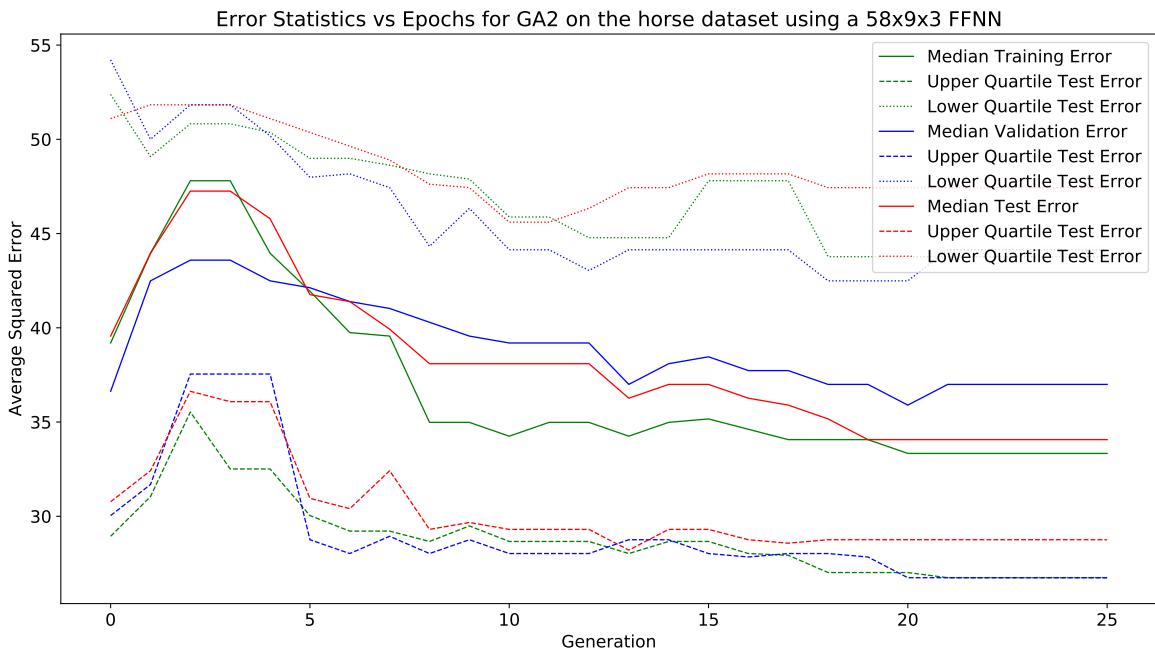
Fig. 392 shows the average and standard deviation of the test, training and validation errors. Fig. 393 shows the median and quartiles of the test, training and validation errors based on the best epoch. Finally, Fig. 394 shows the median and quartiles of the test, training and validation errors per mini-batch.



**Figure 392.** Graph of mean and standard deviation of errors vs epochs



**Figure 393.** Graph of test error vs epochs for the gradient based algorithms



**Figure 394.** Graph of test error vs epochs for the gradient based algorithms

### 16.5.16 GA3

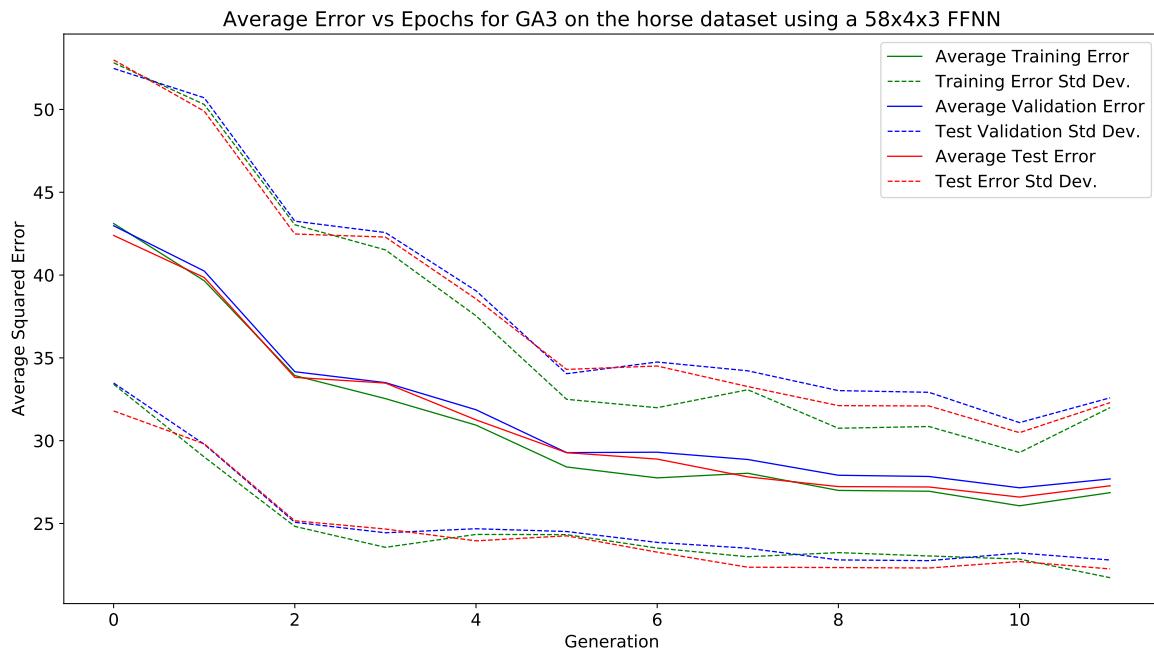
This section displays the results obtained using the GA3 algorithm.

### 16.5.17 horse

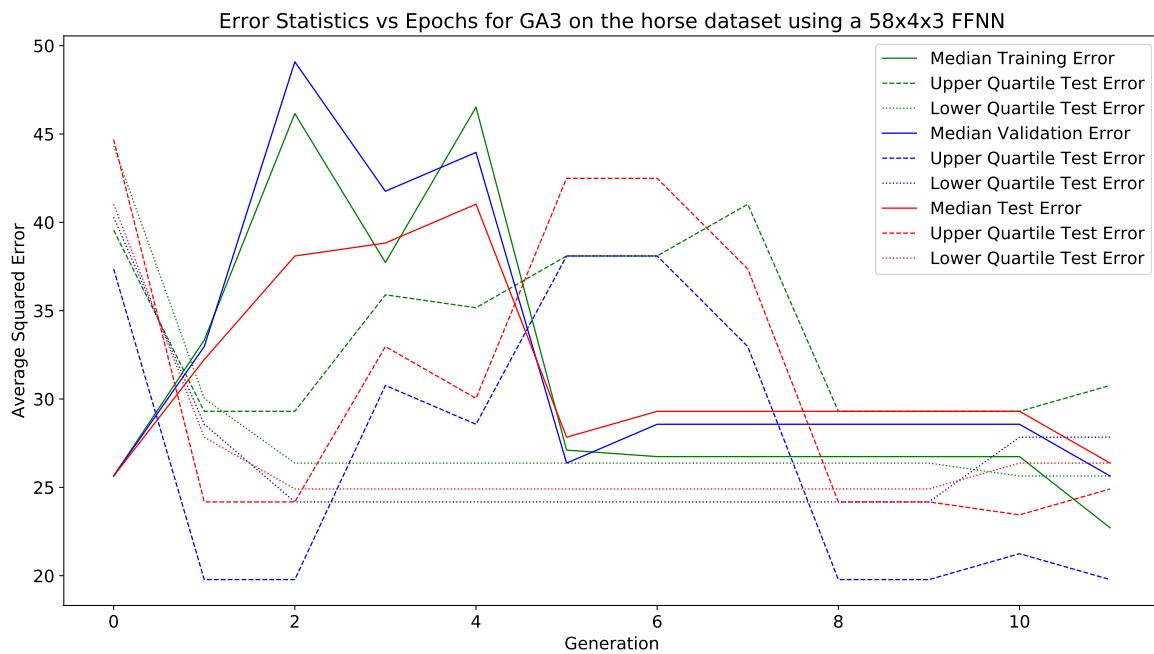
This section displays the results obtained using the horse algorithm.

#### 58 × 4 × 3 Architecture:

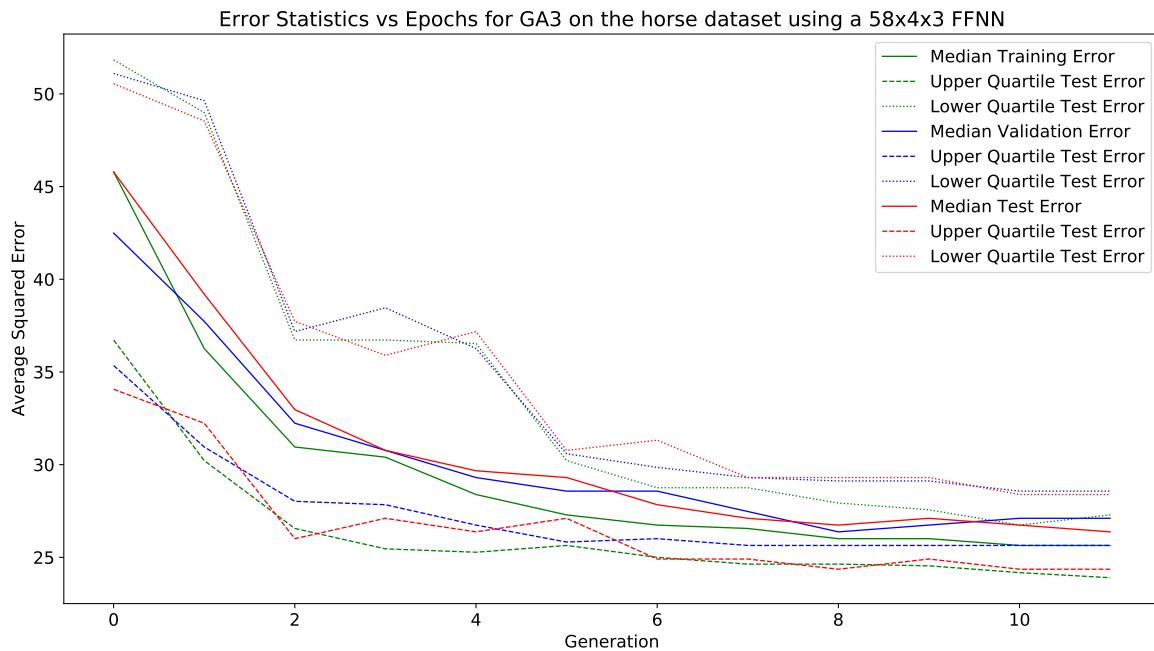
Fig. 395 shows the average and standard deviation of the test, training and validation errors. Fig. 396 shows the median and quartiles of the test, training and validation errors based on the best epoch. Finally, Fig. 397 shows the median and quartiles of the test, training and validation errors per mini-batch.



**Figure 395.** Graph of mean and standard deviation of errors vs epochs



**Figure 396.** Graph of test error vs epochs for the gradient based algorithms



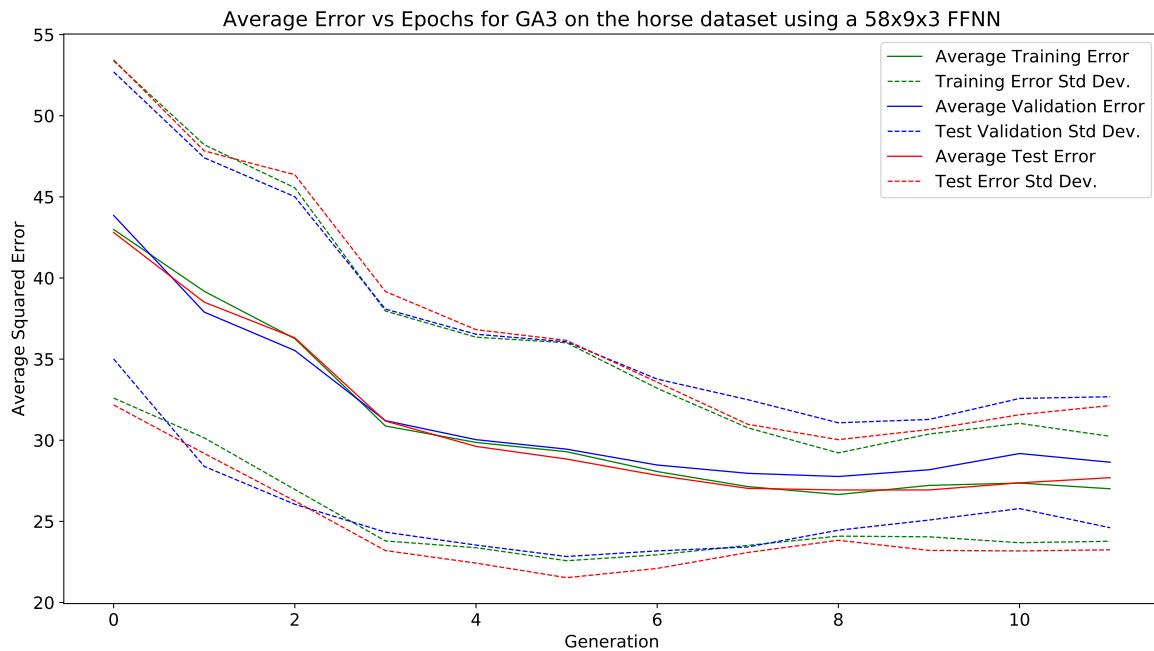
**Figure 397.** Graph of test error vs epochs for the gradient based algorithms

### 16.5.18 horse

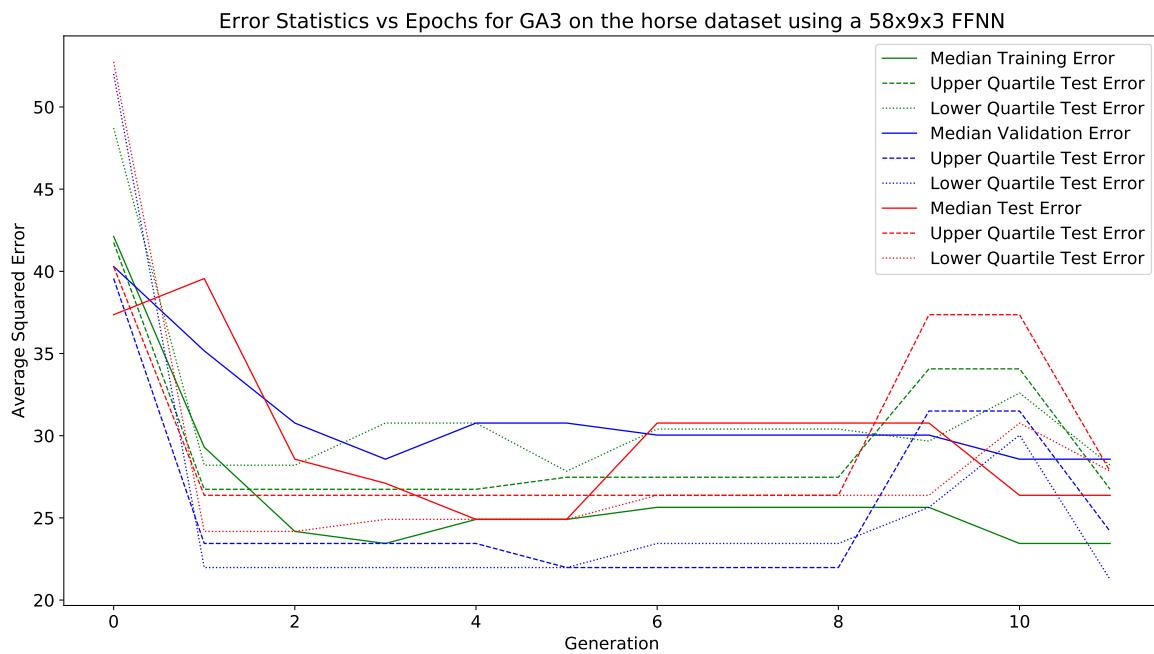
This section displays the results obtained using the horse algorithm.

#### 58 × 9 × 3 Architecture:

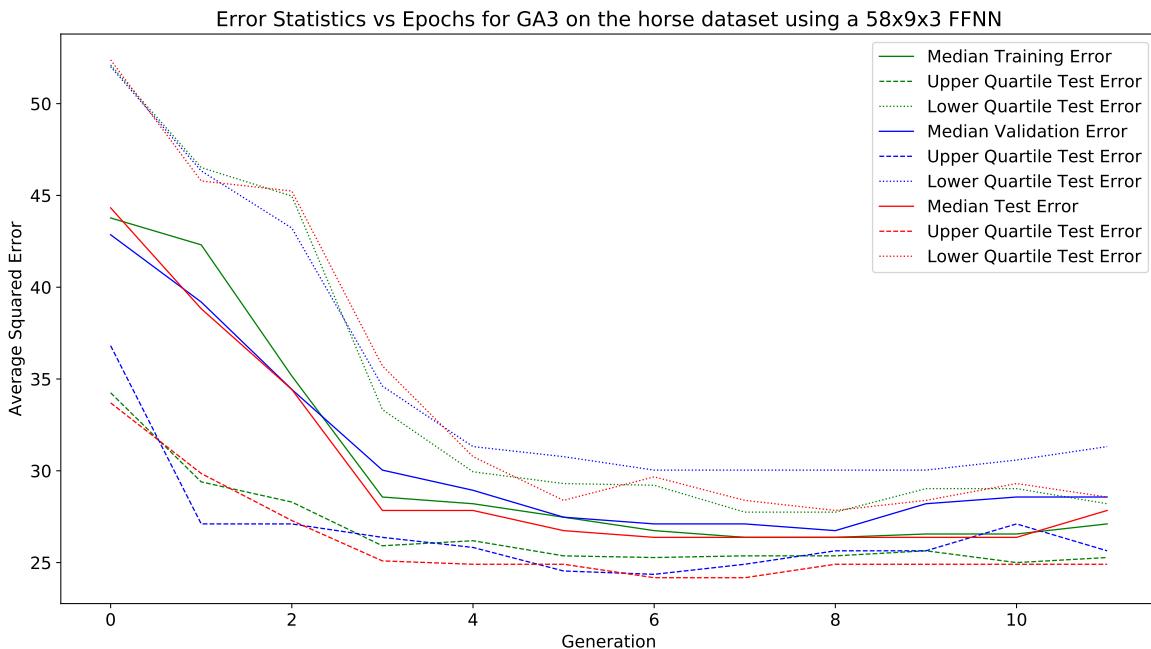
Fig. 398 shows the average and standard deviation of the test, training and validation errors. Fig. 399 shows the median and quartiles of the test, training and validation errors based on the best epoch. Finally, Fig. 400 shows the median and quartiles of the test, training and validation errors per mini-batch.



**Figure 398.** Graph of mean and standard deviation of errors vs epochs



**Figure 399.** Graph of test error vs epochs for the gradient based algorithms



**Figure 400.** Graph of test error vs epochs for the gradient based algorithms

### 16.5.19 iRPROP-

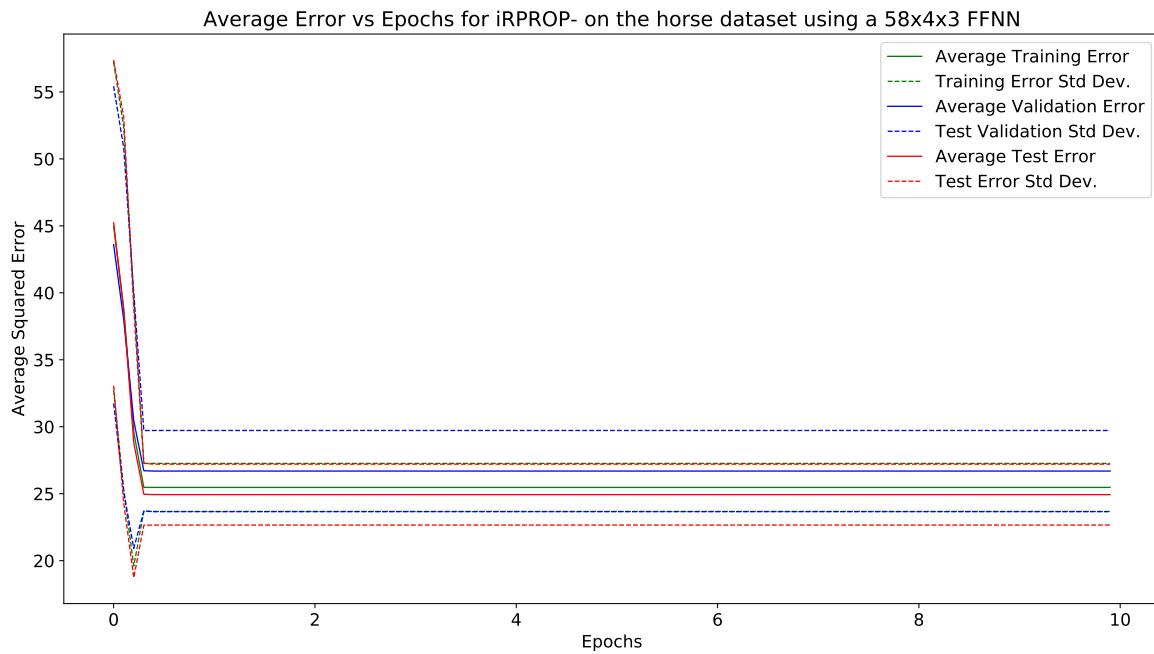
This section displays the results obtained using the iRPROP- algorithm.

### 16.5.20 horse

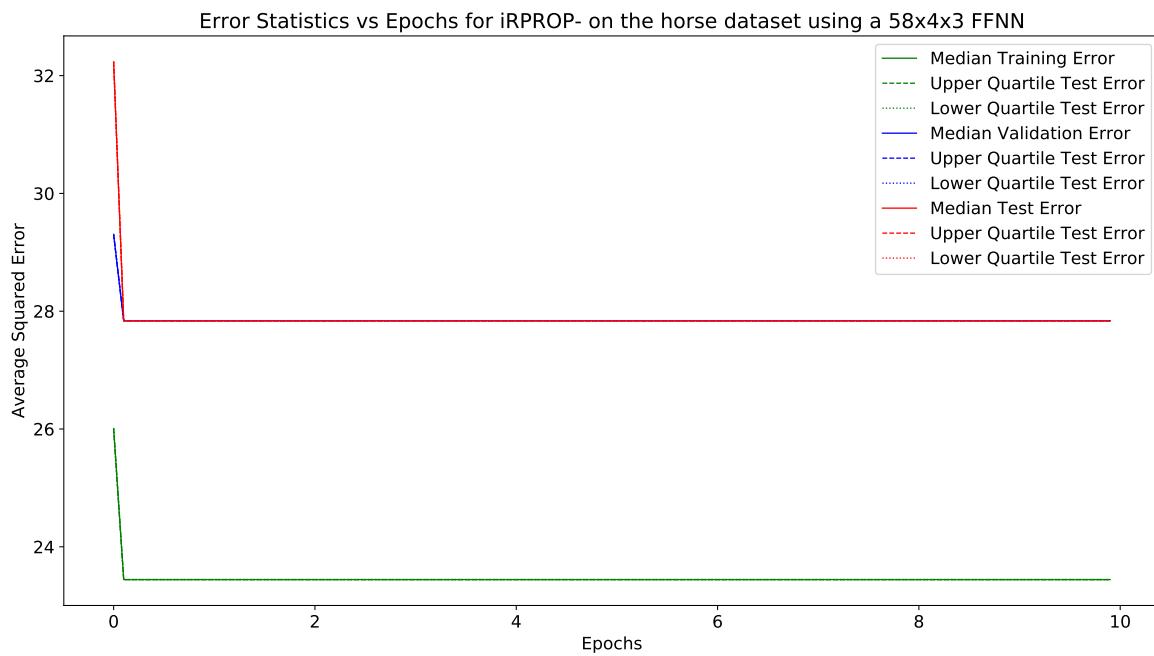
This section displays the results obtained using the horse algorithm.

#### 58 × 4 × 3 Architecture:

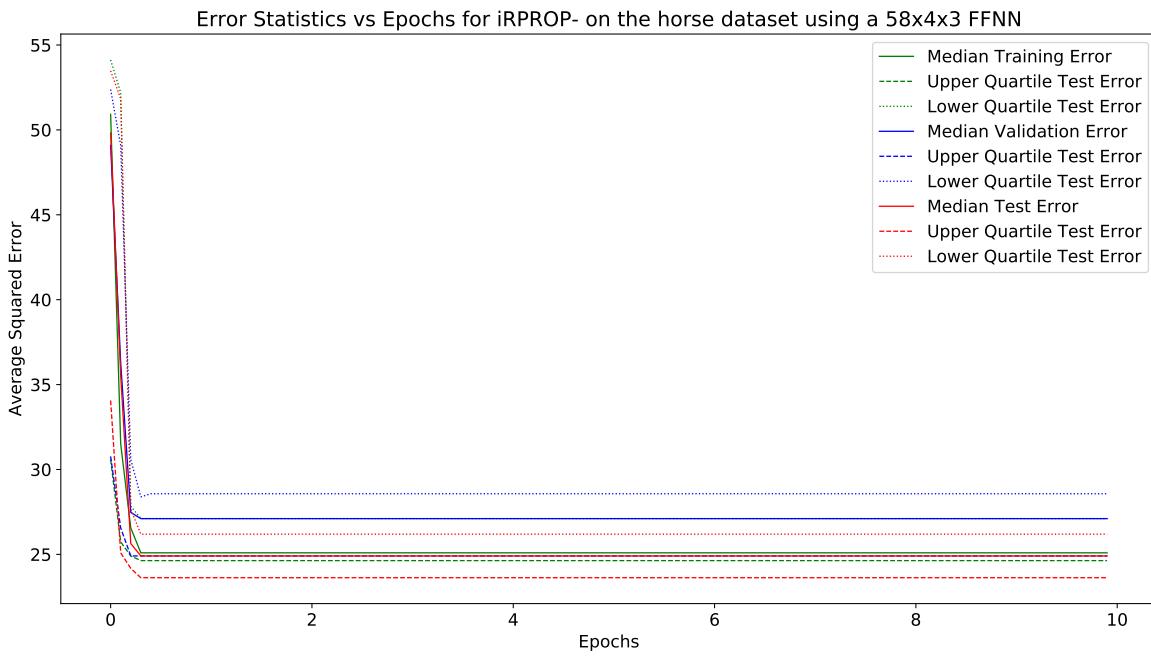
Fig. 401 shows the average and standard deviation of the test, training and validation errors. Fig. 402 shows the median and quartiles of the test, training and validation errors based on the best epoch. Finally, Fig. 403 shows the median and quartiles of the test, training and validation errors per mini-batch.



**Figure 401.** Graph of mean and standard deviation of errors vs epochs



**Figure 402.** Graph of test error vs epochs for the gradient based algorithms



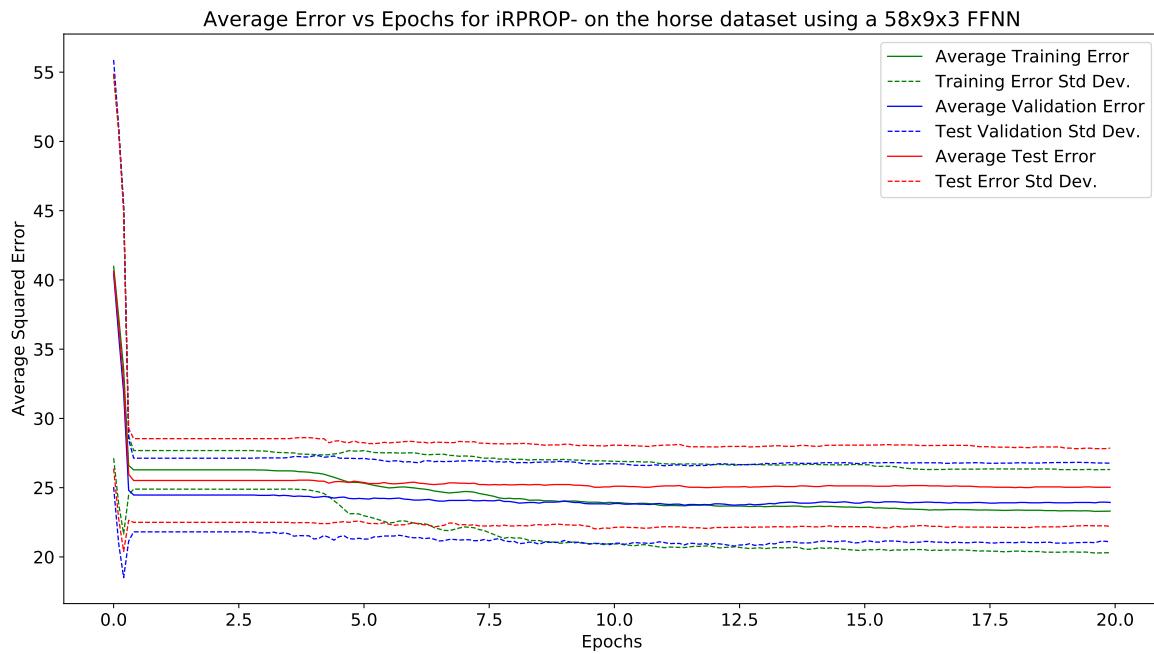
**Figure 403.** Graph of test error vs epochs for the gradient based algorithms

### 16.5.21 horse

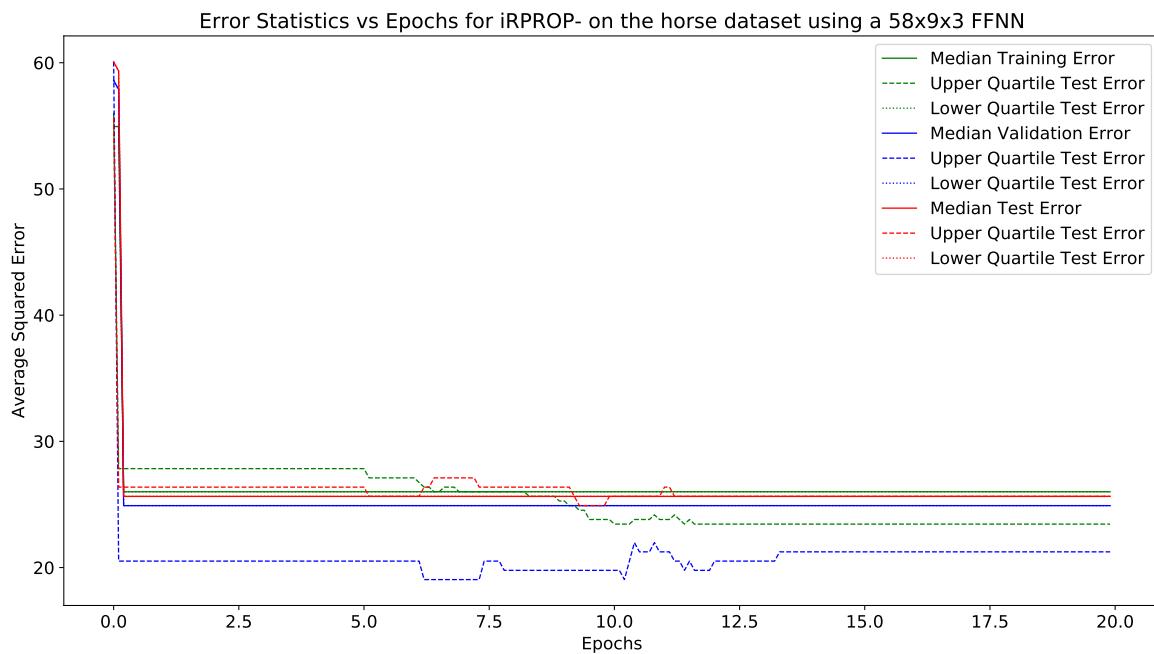
This section displays the results obtained using the horse algorithm.

#### 58 × 9 × 3 Architecture:

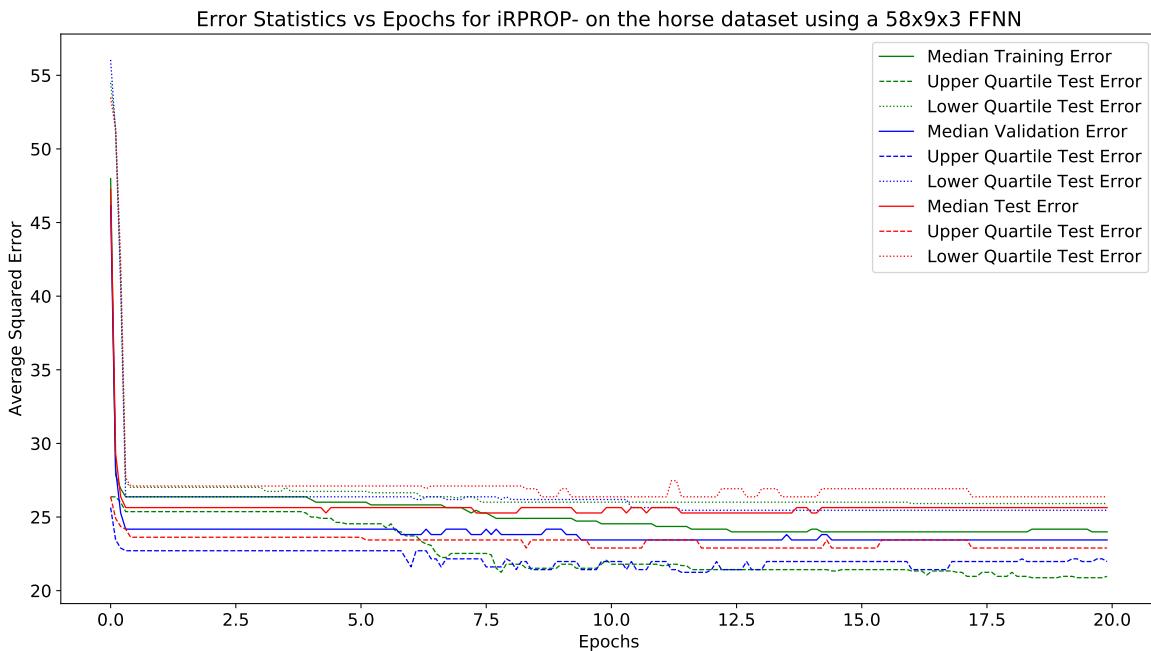
Fig. 404 shows the average and standard deviation of the test, training and validation errors. Fig. 405 shows the median and quartiles of the test, training and validation errors based on the best epoch. Finally, Fig. 406 shows the median and quartiles of the test, training and validation errors per mini-batch.



**Figure 404.** Graph of mean and standard deviation of errors vs epochs



**Figure 405.** Graph of test error vs epochs for the gradient based algorithms



**Figure 406.** Graph of test error vs epochs for the gradient based algorithms

### 16.5.22 *iRPROP+*

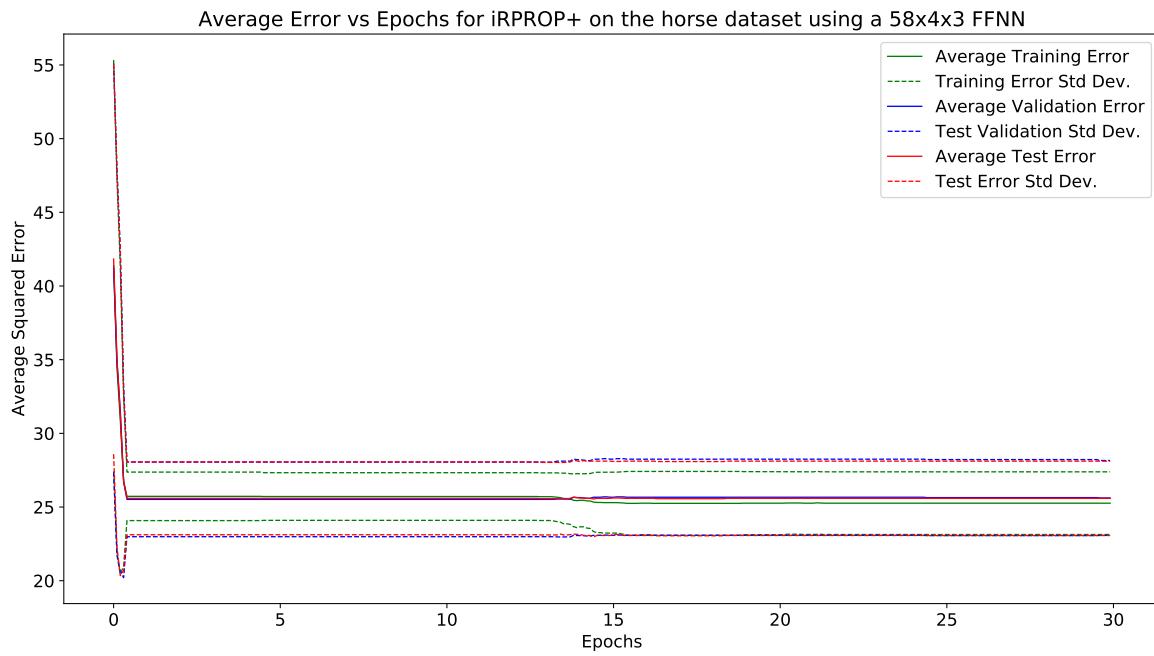
This section displays the results obtained using the iRPROP+ algorithm.

### 16.5.23 *horse*

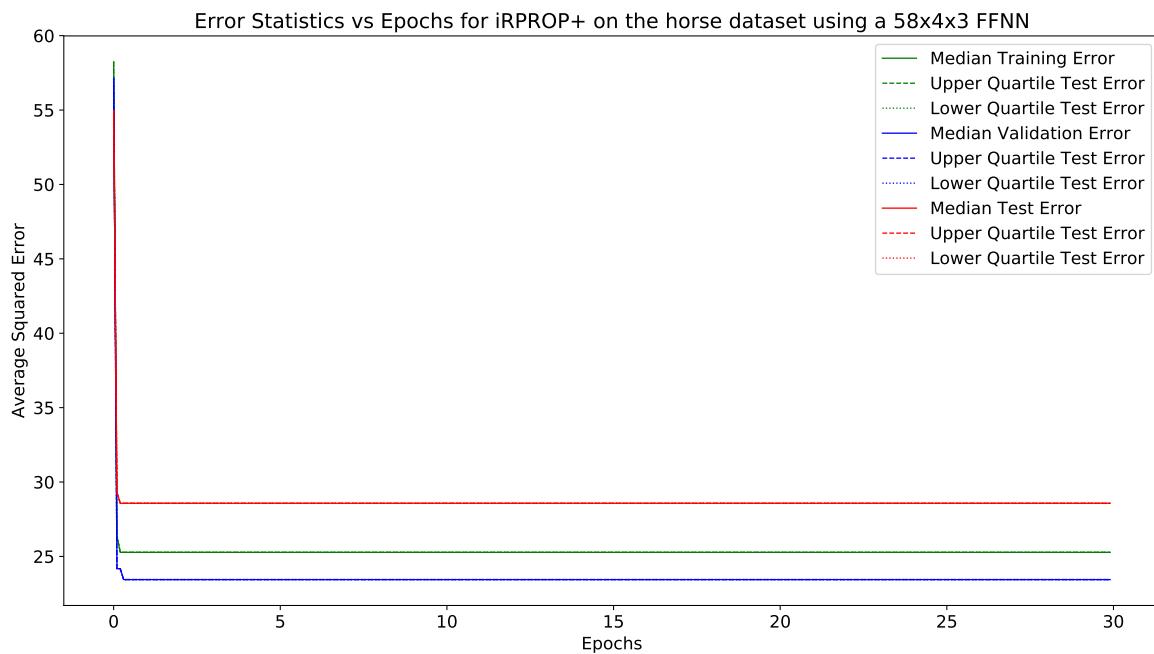
This section displays the results obtained using the horse algorithm.

#### 58 × 4 × 3 Architecture:

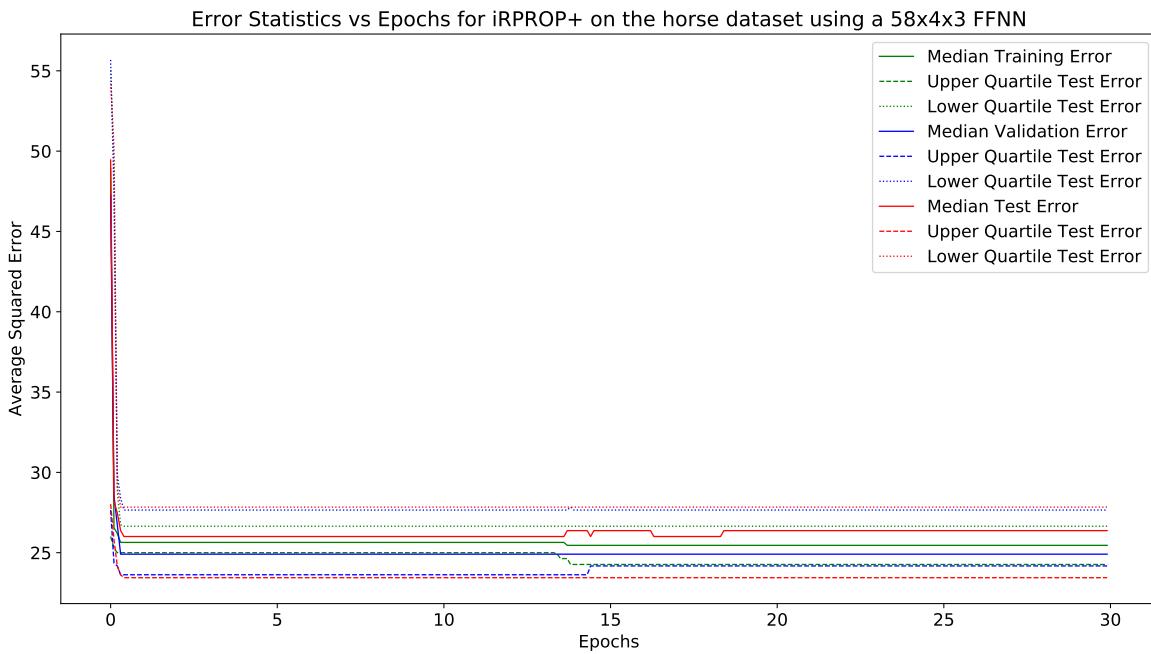
Fig. 407 shows the average and standard deviation of the test, training and validation errors. Fig. 408 shows the median and quartiles of the test, training and validation errors based on the best epoch. Finally, Fig. 409 shows the median and quartiles of the test, training and validation errors per mini-batch.



**Figure 407.** Graph of mean and standard deviation of errors vs epochs



**Figure 408.** Graph of test error vs epochs for the gradient based algorithms



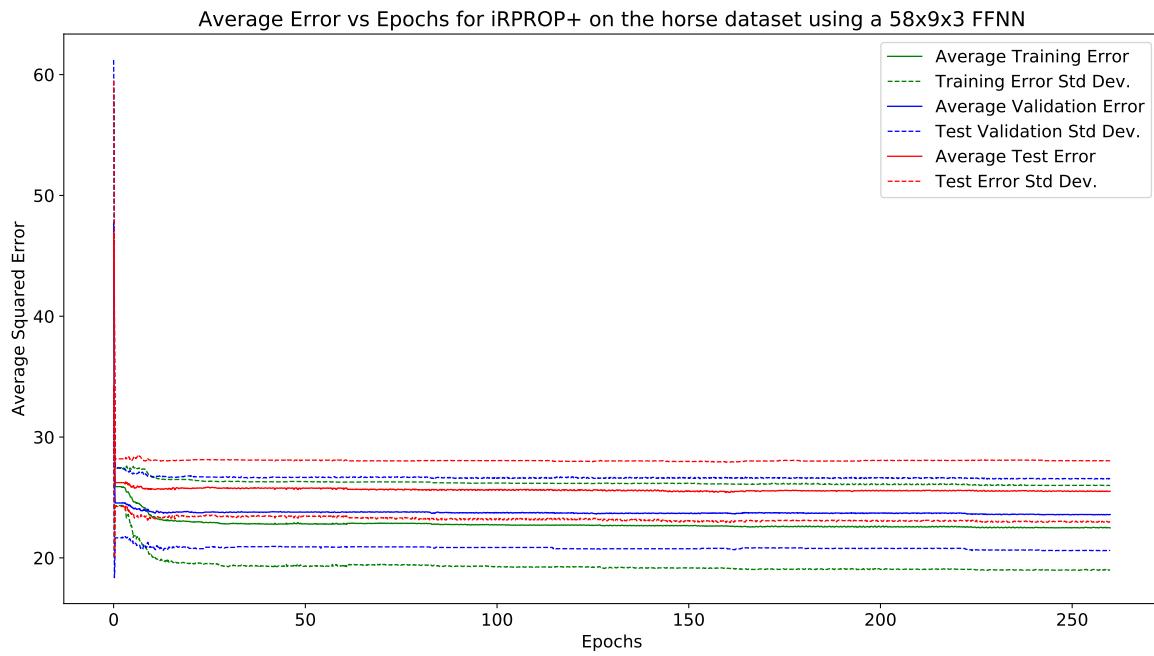
**Figure 409.** Graph of test error vs epochs for the gradient based algorithms

### 16.5.24 horse

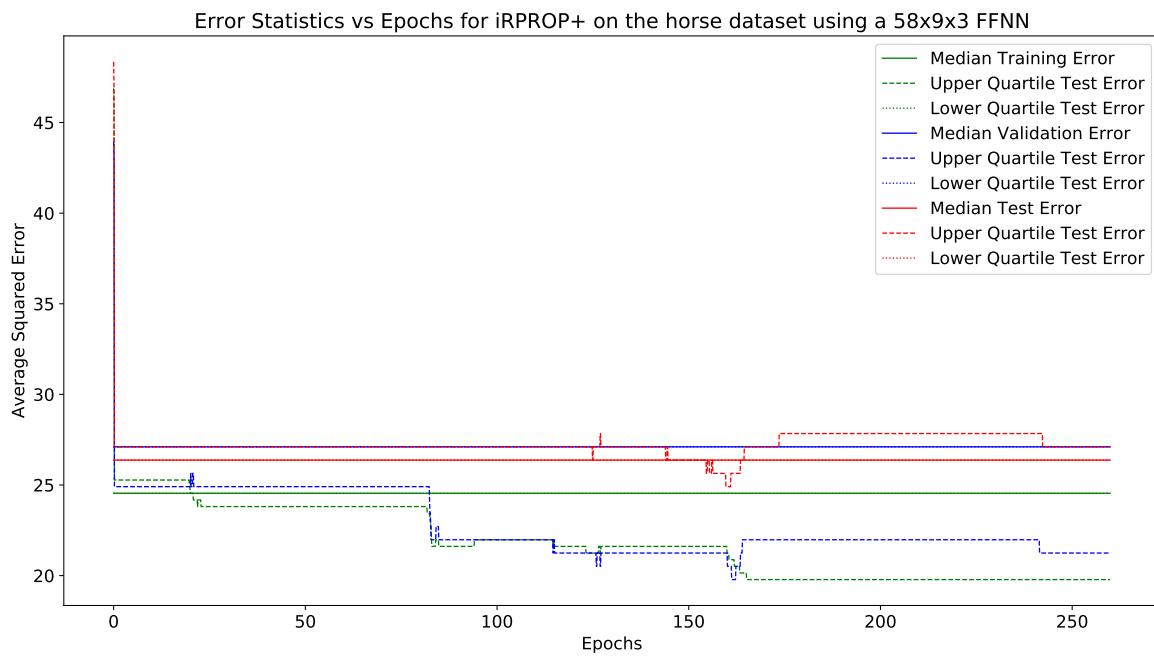
This section displays the results obtained using the horse algorithm.

#### 58 × 9 × 3 Architecture:

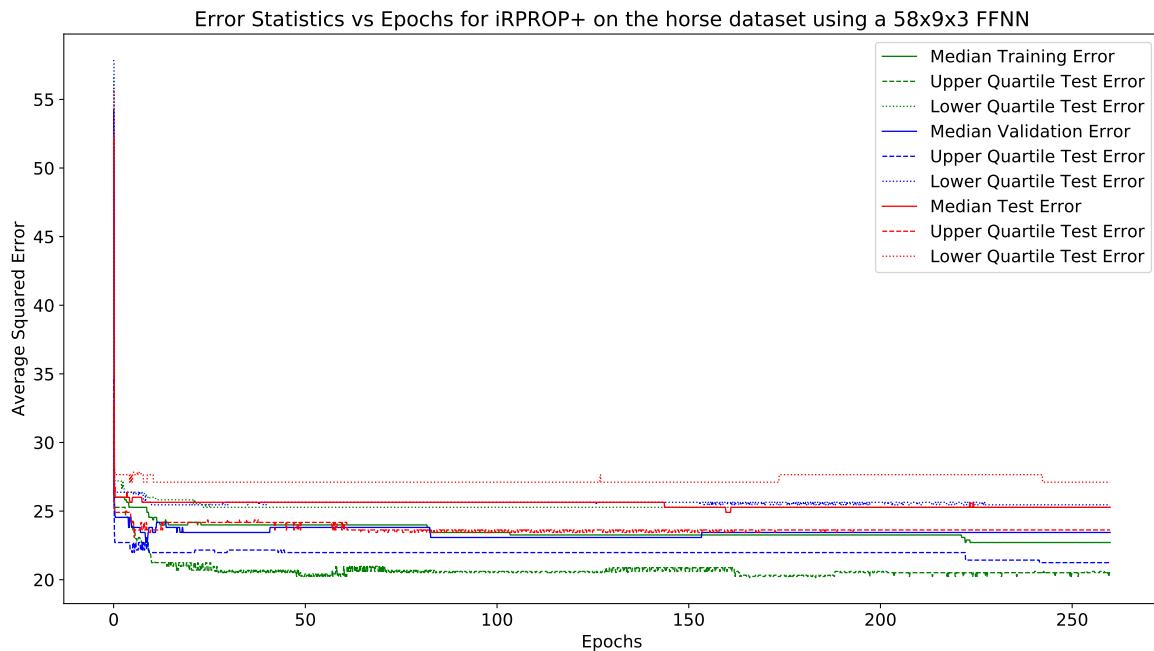
Fig. 410 shows the average and standard deviation of the test, training and validation errors. Fig. 411 shows the median and quartiles of the test, training and validation errors based on the best epoch. Finally, Fig. 412 shows the median and quartiles of the test, training and validation errors per mini-batch.



**Figure 410.** Graph of mean and standard deviation of errors vs epochs



**Figure 411.** Graph of test error vs epochs for the gradient based algorithms



**Figure 412.** Graph of test error vs epochs for the gradient based algorithms

### 16.5.25 PSO

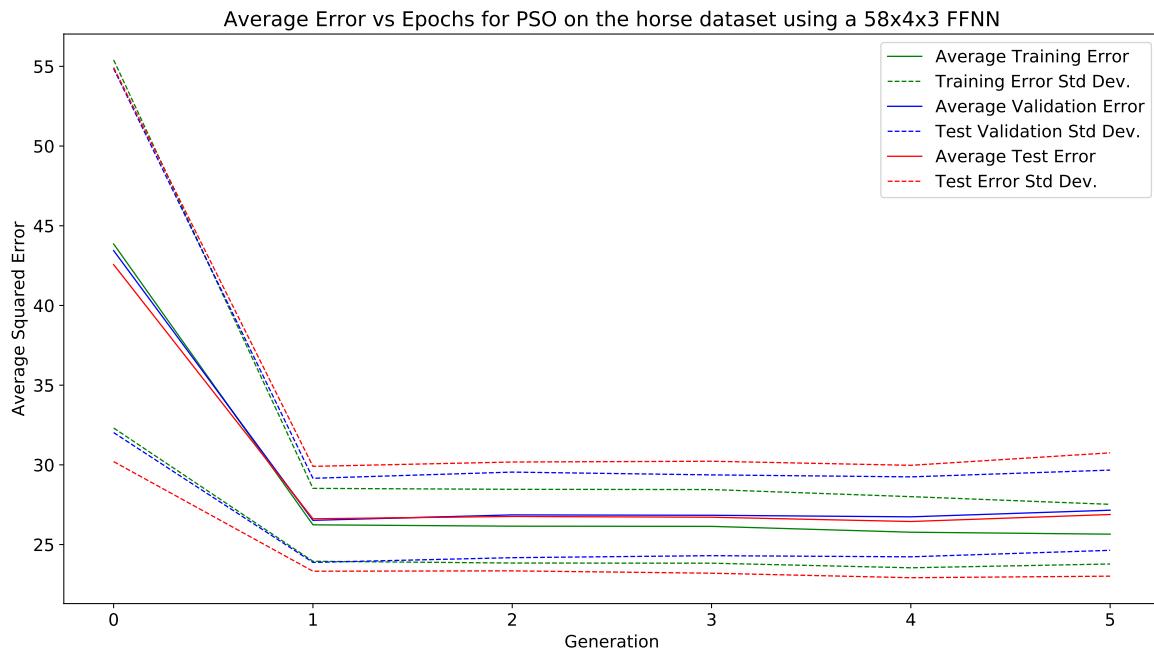
This section displays the results obtained using the PSO algorithm.

### 16.5.26 horse

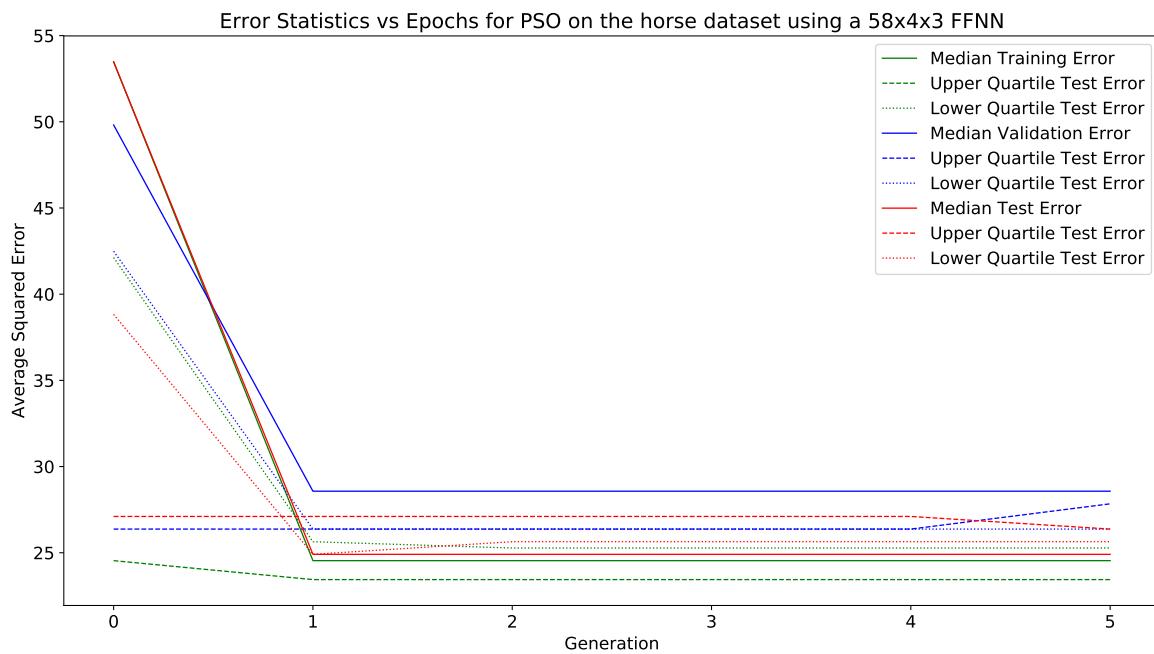
This section displays the results obtained using the horse algorithm.

#### 58 × 4 × 3 Architecture:

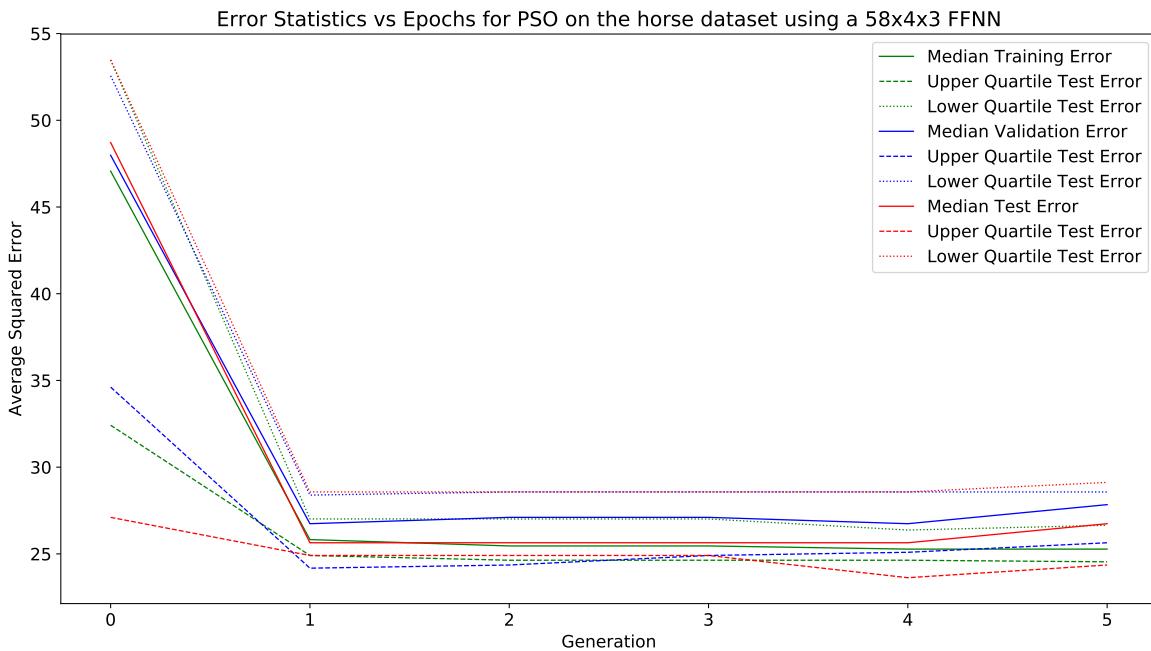
Fig. 413 shows the average and standard deviation of the test, training and validation errors. Fig. 414 shows the median and quartiles of the test, training and validation errors based on the best epoch. Finally, Fig. 415 shows the median and quartiles of the test, training and validation errors per mini-batch.



**Figure 413.** Graph of mean and standard deviation of errors vs epochs



**Figure 414.** Graph of test error vs epochs for the gradient based algorithms



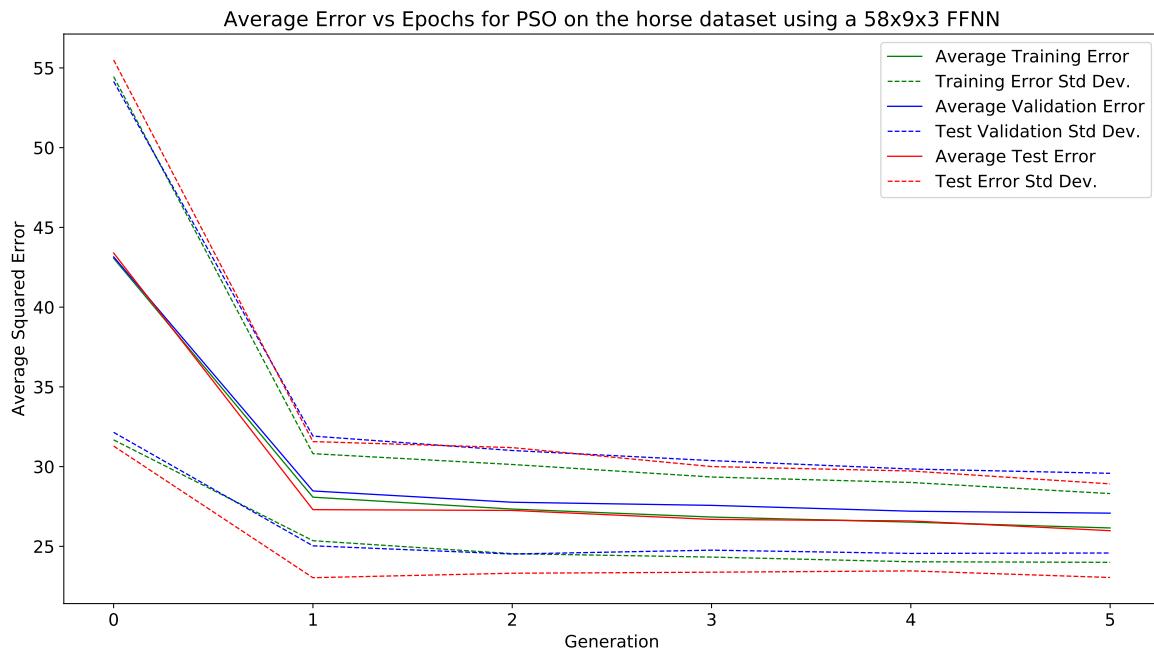
**Figure 415.** Graph of test error vs epochs for the gradient based algorithms

### 16.5.27 horse

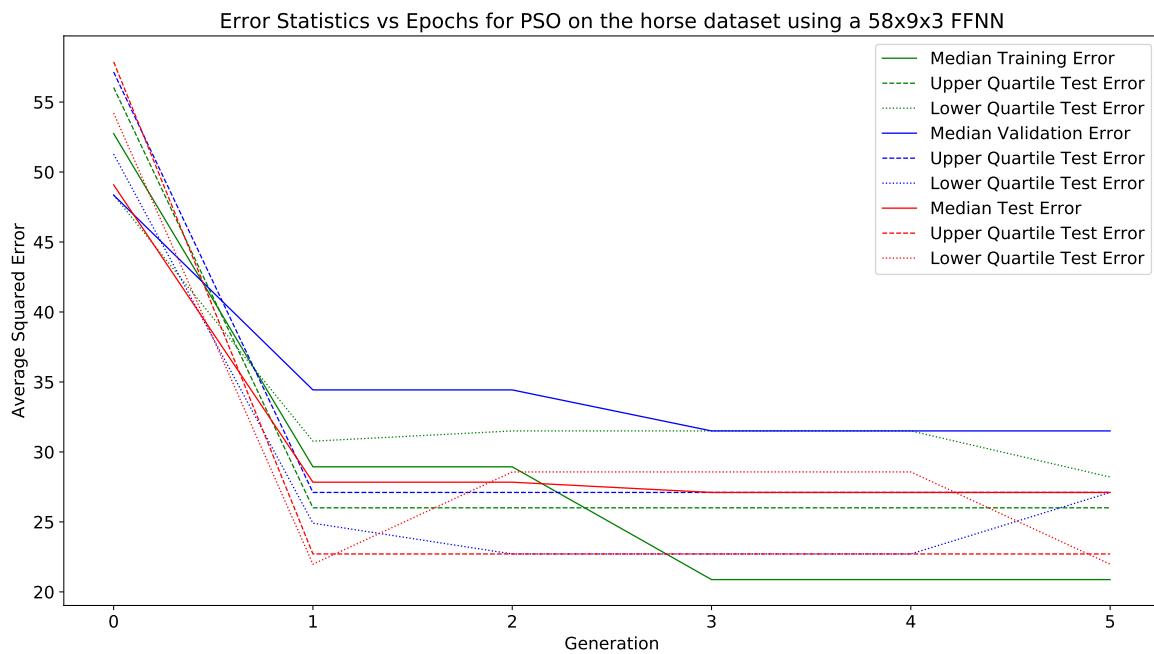
This section displays the results obtained using the horse algorithm.

#### 58 × 9 × 3 Architecture:

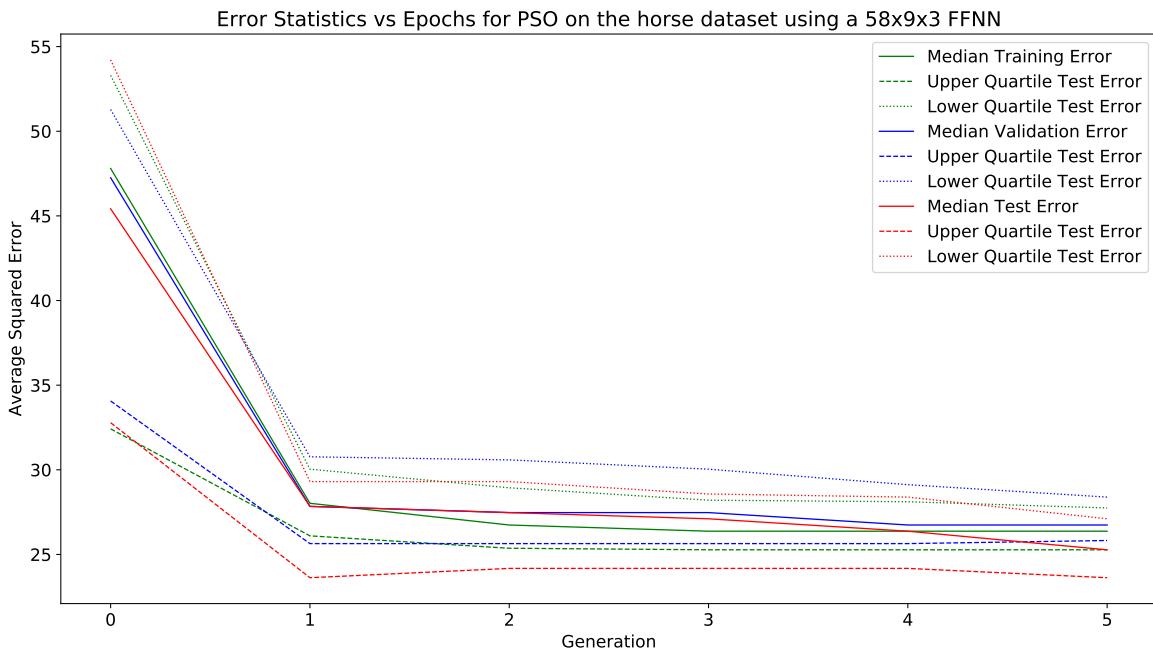
Fig. 416 shows the average and standard deviation of the test, training and validation errors. Fig. 417 shows the median and quartiles of the test, training and validation errors based on the best epoch. Finally, Fig. 418 shows the median and quartiles of the test, training and validation errors per mini-batch.



**Figure 416.** Graph of mean and standard deviation of errors vs epochs



**Figure 417.** Graph of test error vs epochs for the gradient based algorithms



**Figure 418.** Graph of test error vs epochs for the gradient based algorithms

### 16.5.28 QuickProp

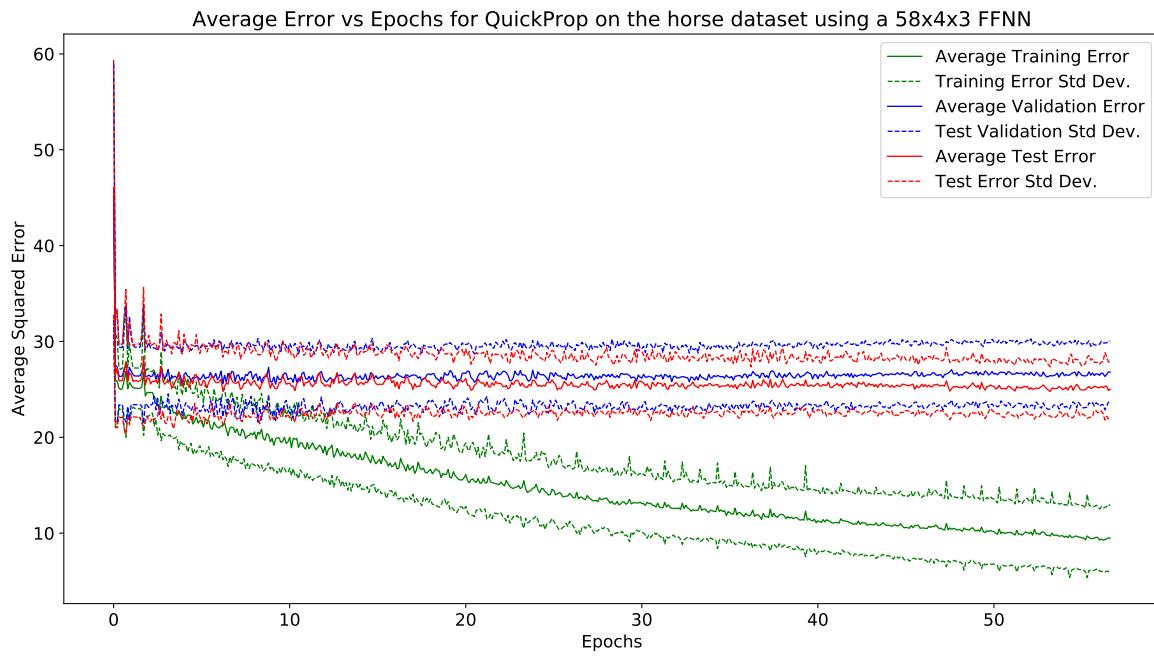
This section displays the results obtained using the QuickProp algorithm.

### 16.5.29 horse

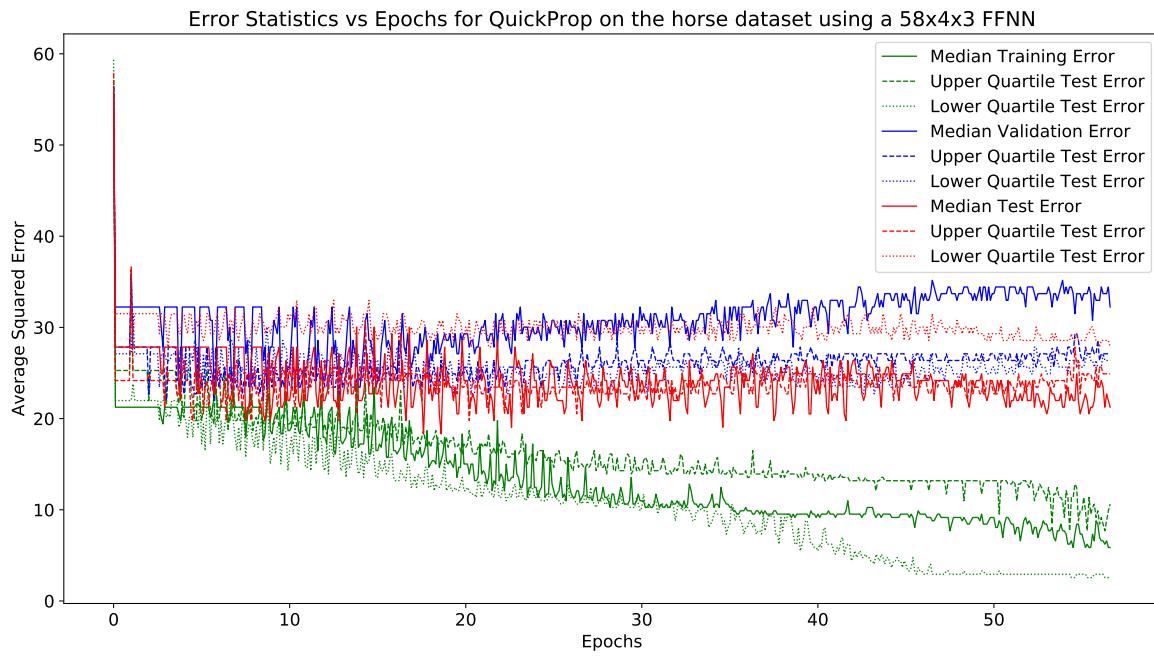
This section displays the results obtained using the horse algorithm.

#### 58 × 4 × 3 Architecture:

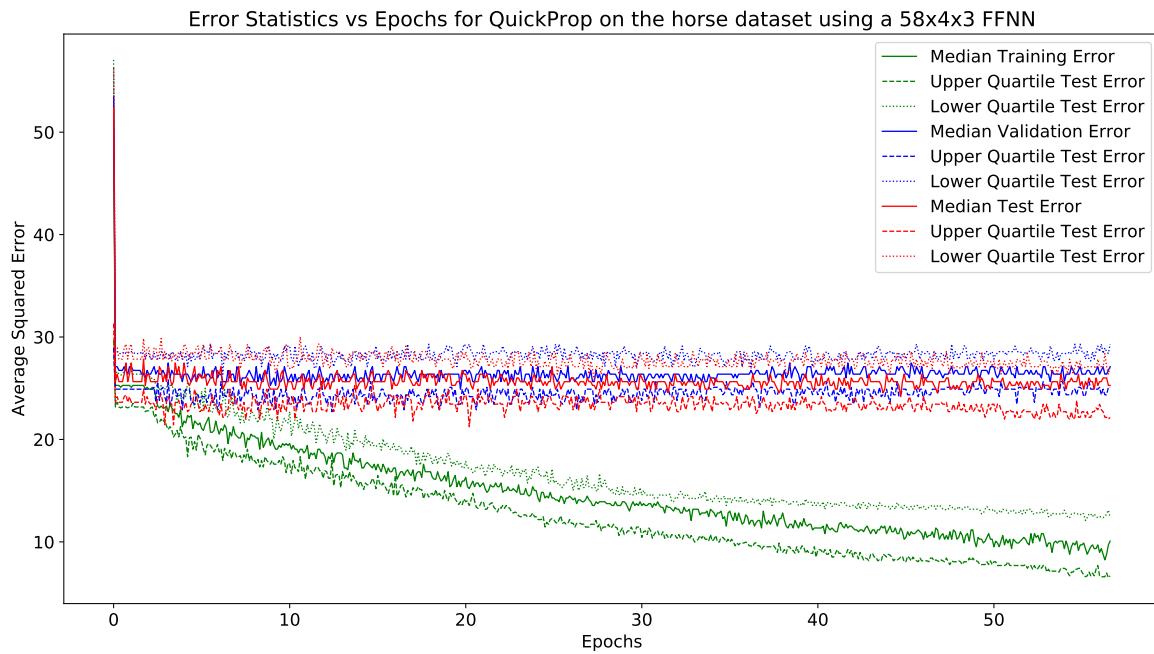
Fig. 419 shows the average and standard deviation of the test, training and validation errors. Fig. 420 shows the median and quartiles of the test, training and validation errors based on the best epoch. Finally, Fig. 421 shows the median and quartiles of the test, training and validation errors per mini-batch.



**Figure 419.** Graph of mean and standard deviation of errors vs epochs



**Figure 420.** Graph of test error vs epochs for the gradient based algorithms



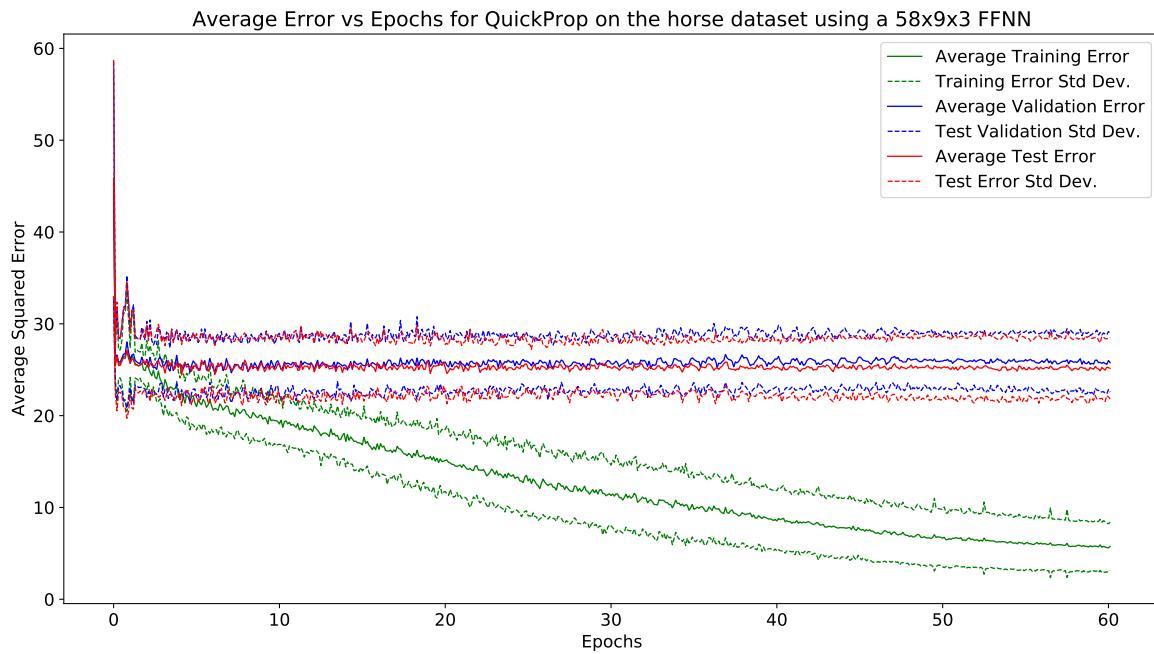
**Figure 421.** Graph of test error vs epochs for the gradient based algorithms

### 16.5.30 horse

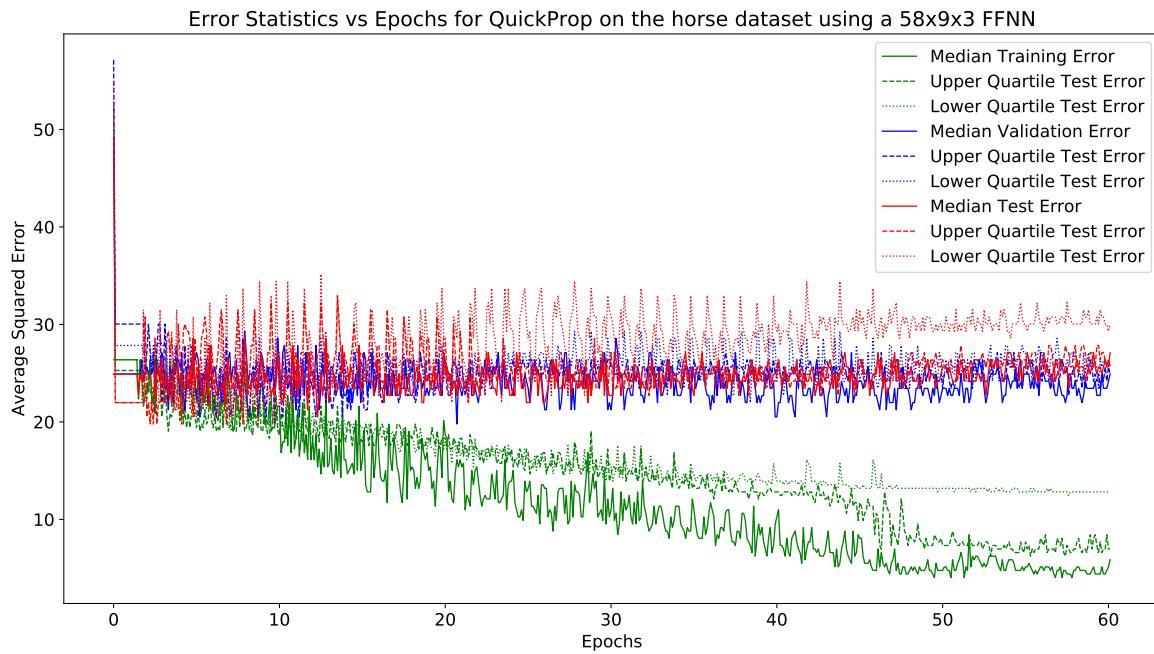
This section displays the results obtained using the horse algorithm.

#### 58 × 9 × 3 Architecture:

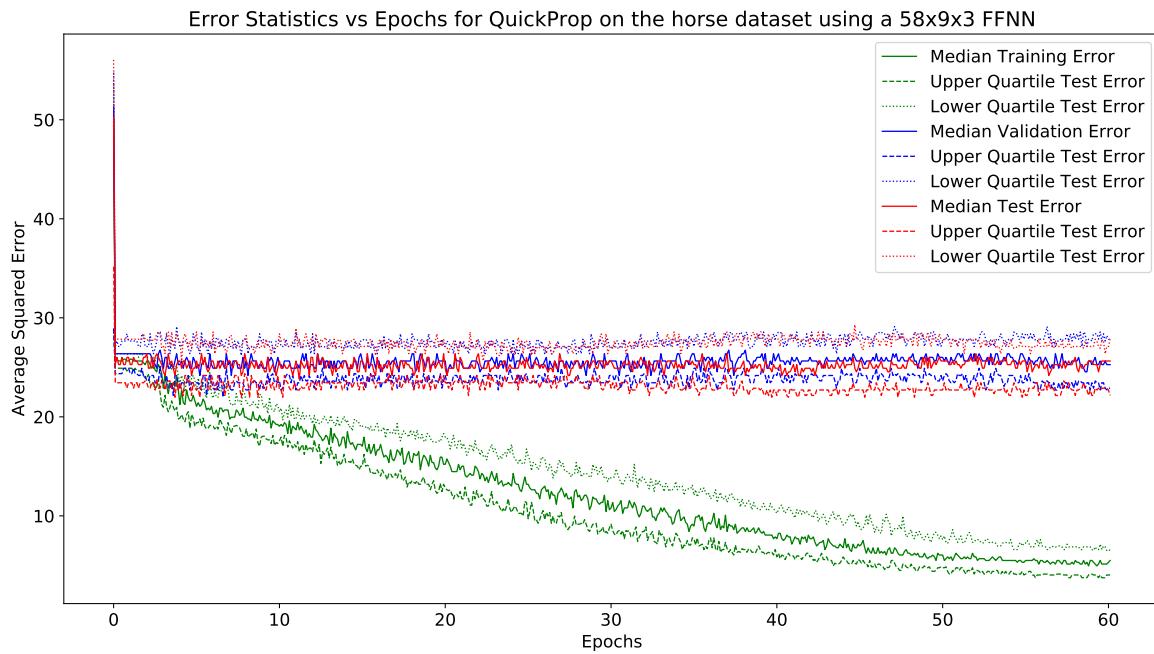
Fig. 422 shows the average and standard deviation of the test, training and validation errors. Fig. 423 shows the median and quartiles of the test, training and validation errors based on the best epoch. Finally, Fig. 424 shows the median and quartiles of the test, training and validation errors per mini-batch.



**Figure 422.** Graph of mean and standard deviation of errors vs epochs



**Figure 423.** Graph of test error vs epochs for the gradient based algorithms



**Figure 424.** Graph of test error vs epochs for the gradient based algorithms

### 16.5.31 RPROP-

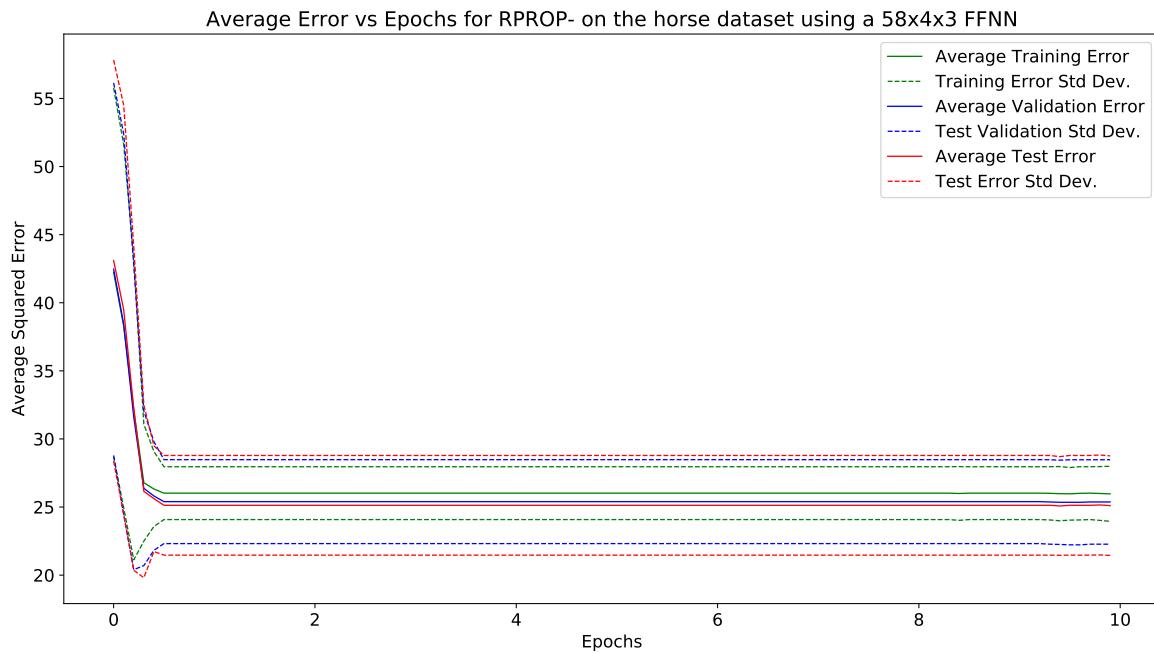
This section displays the results obtained using the RPROP- algorithm.

### 16.5.32 horse

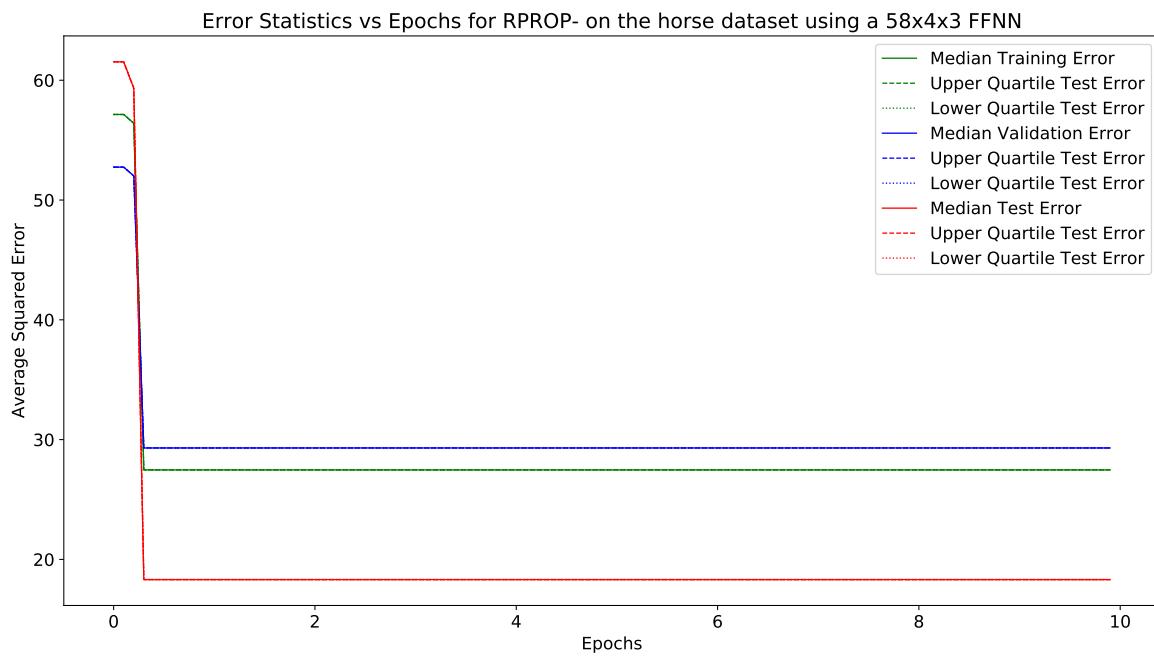
This section displays the results obtained using the horse algorithm.

#### 58 × 4 × 3 Architecture:

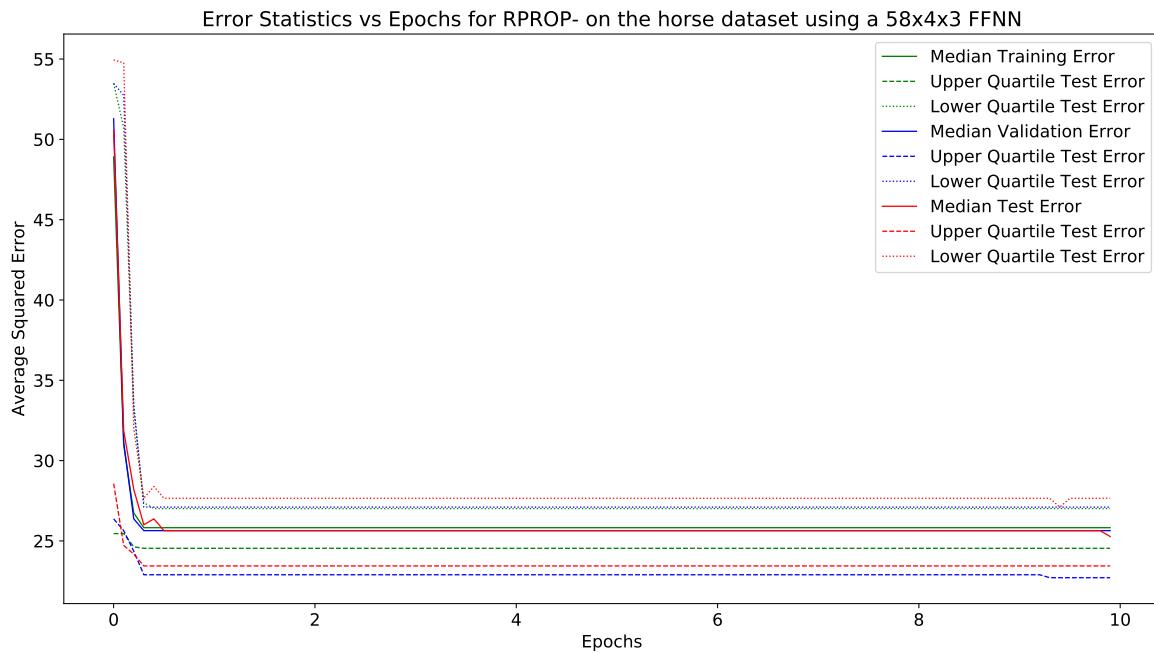
Fig. 425 shows the average and standard deviation of the test, training and validation errors. Fig. 426 shows the median and quartiles of the test, training and validation errors based on the best epoch. Finally, Fig. 427 shows the median and quartiles of the test, training and validation errors per mini-batch.



**Figure 425.** Graph of mean and standard deviation of errors vs epochs



**Figure 426.** Graph of test error vs epochs for the gradient based algorithms



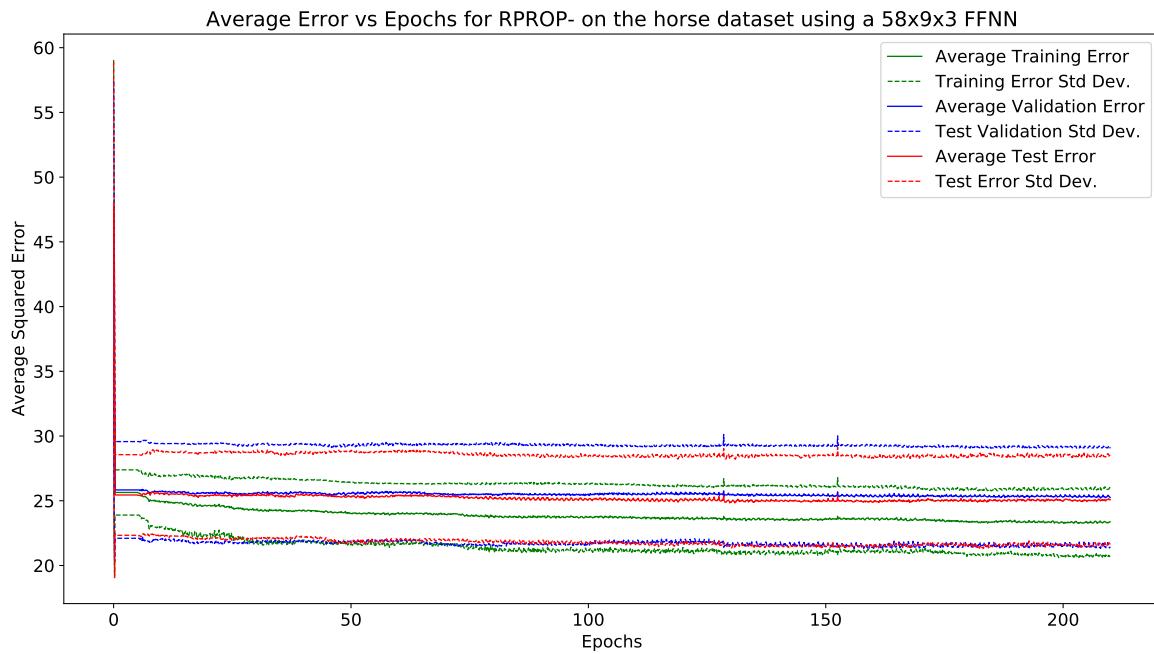
**Figure 427.** Graph of test error vs epochs for the gradient based algorithms

### 16.5.33 horse

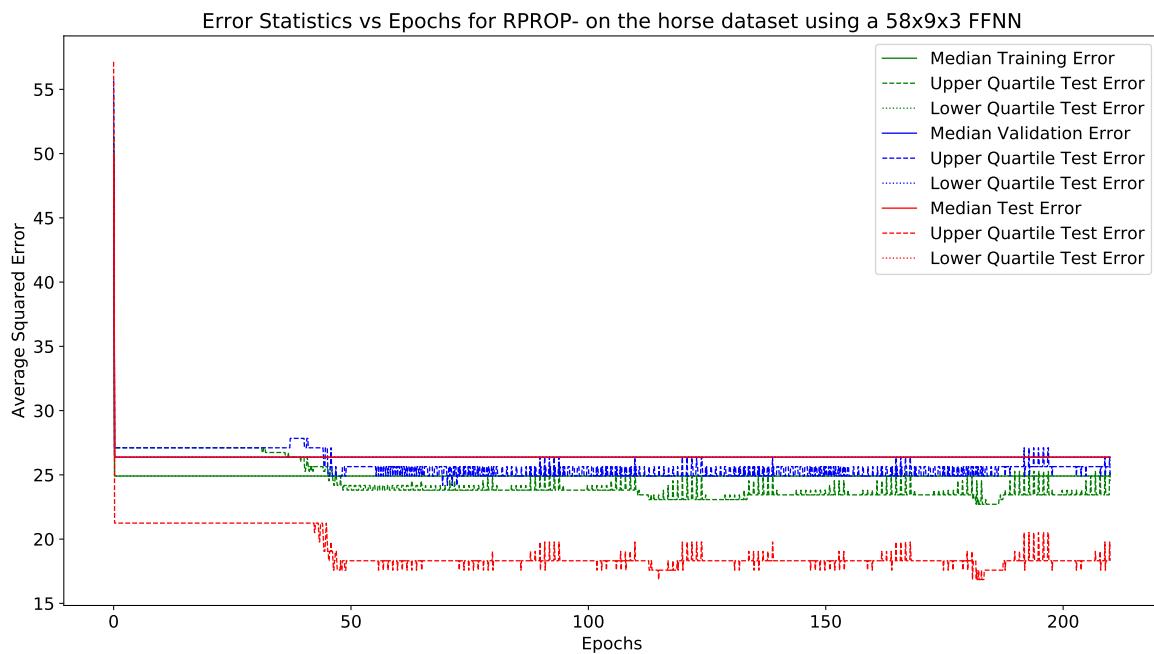
This section displays the results obtained using the horse algorithm.

#### 58 × 9 × 3 Architecture:

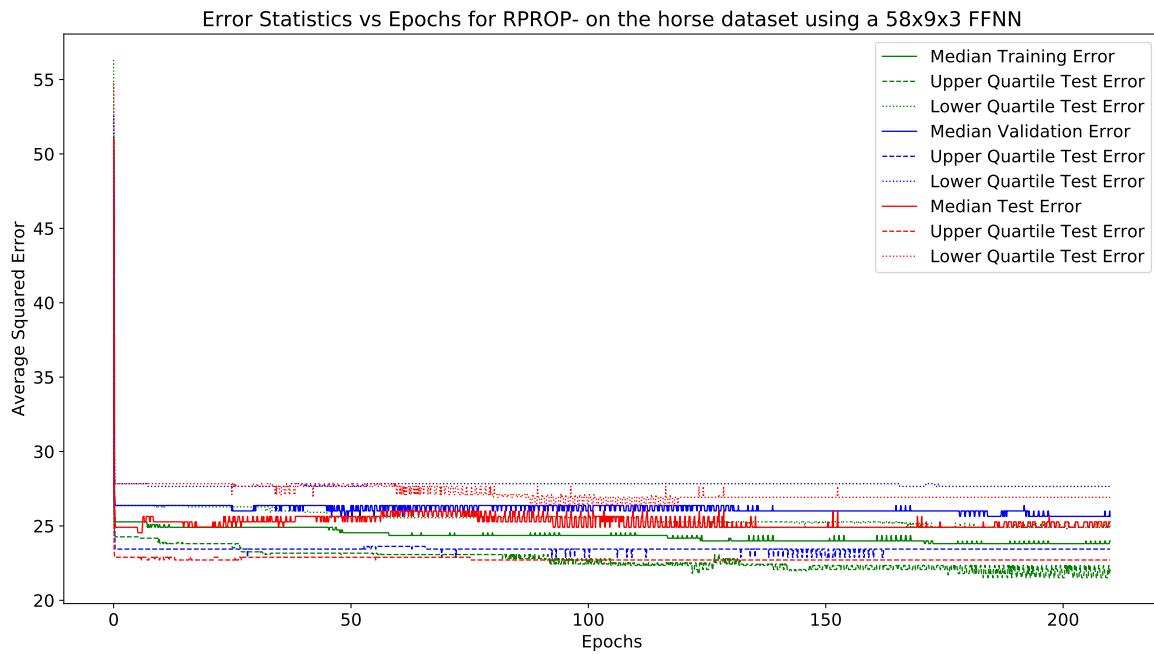
Fig. 428 shows the average and standard deviation of the test, training and validation errors. Fig. 429 shows the median and quartiles of the test, training and validation errors based on the best epoch. Finally, Fig. 430 shows the median and quartiles of the test, training and validation errors per mini-batch.



**Figure 428.** Graph of mean and standard deviation of errors vs epochs



**Figure 429.** Graph of test error vs epochs for the gradient based algorithms



**Figure 430.** Graph of test error vs epochs for the gradient based algorithms

### 16.5.34 RPROP+

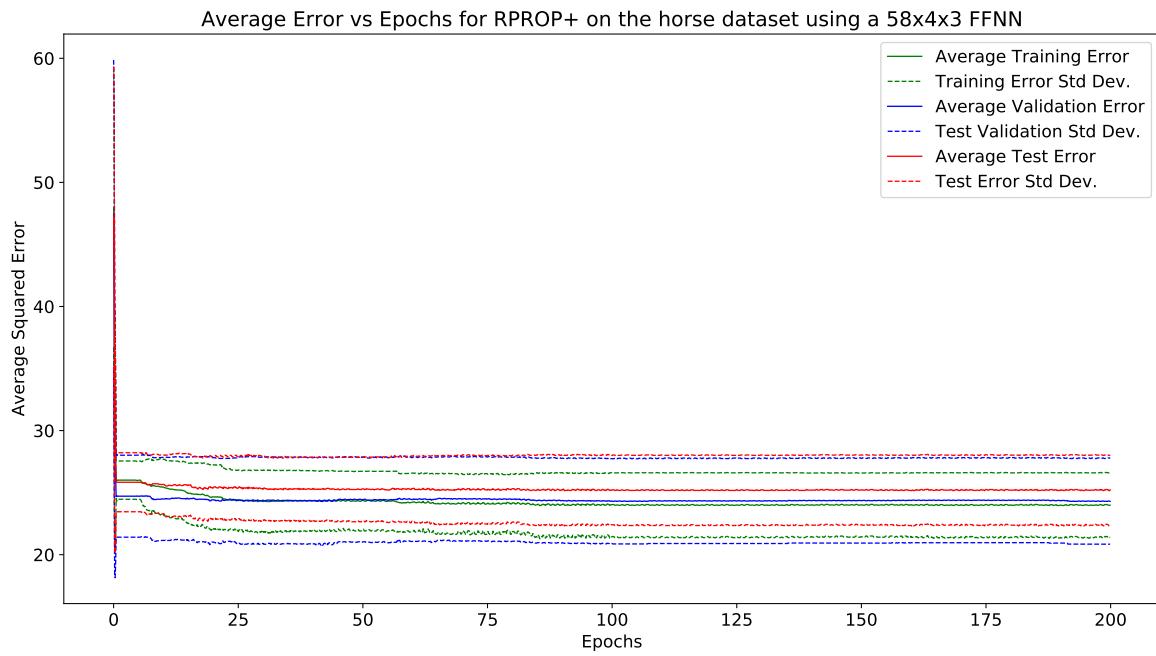
This section displays the results obtained using the RPROP+ algorithm.

### 16.5.35 horse

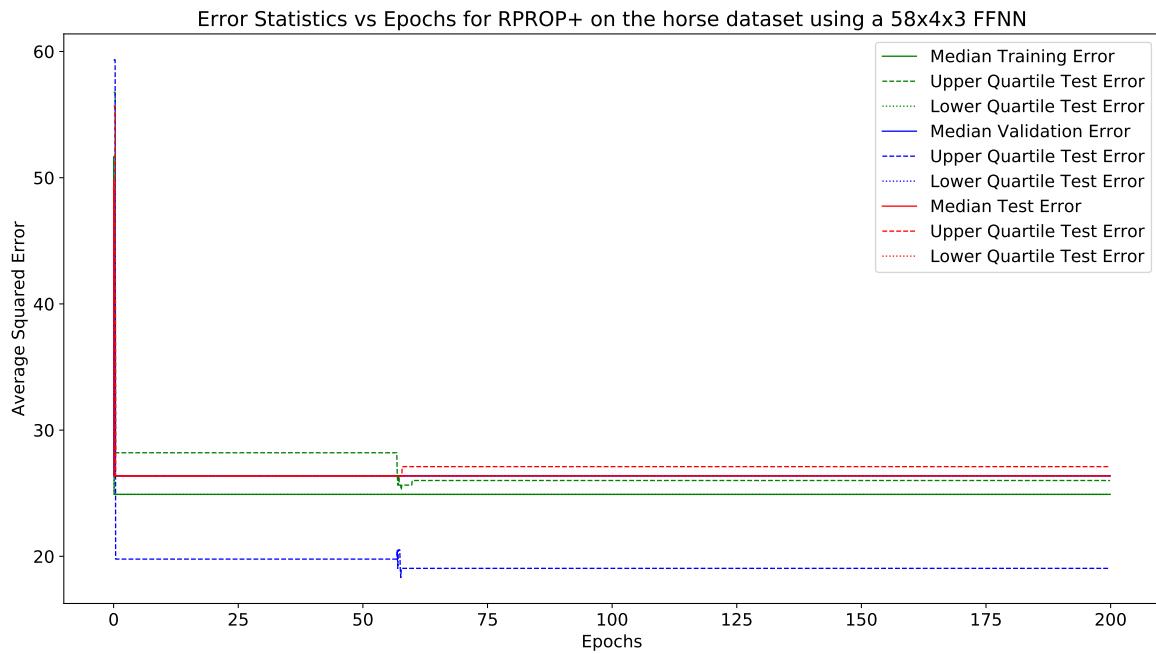
This section displays the results obtained using the horse algorithm.

#### 58 × 4 × 3 Architecture:

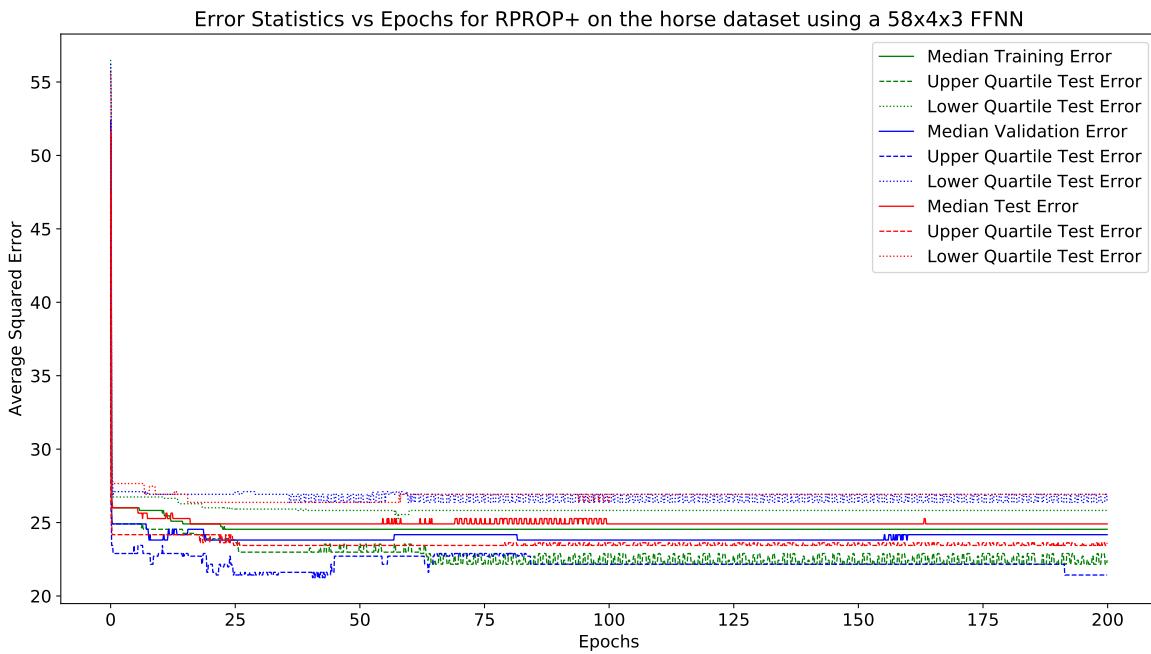
Fig. 431 shows the average and standard deviation of the test, training and validation errors. Fig. 432 shows the median and quartiles of the test, training and validation errors based on the best epoch. Finally, Fig. 433 shows the median and quartiles of the test, training and validation errors per mini-batch.



**Figure 431.** Graph of mean and standard deviation of errors vs epochs



**Figure 432.** Graph of test error vs epochs for the gradient based algorithms



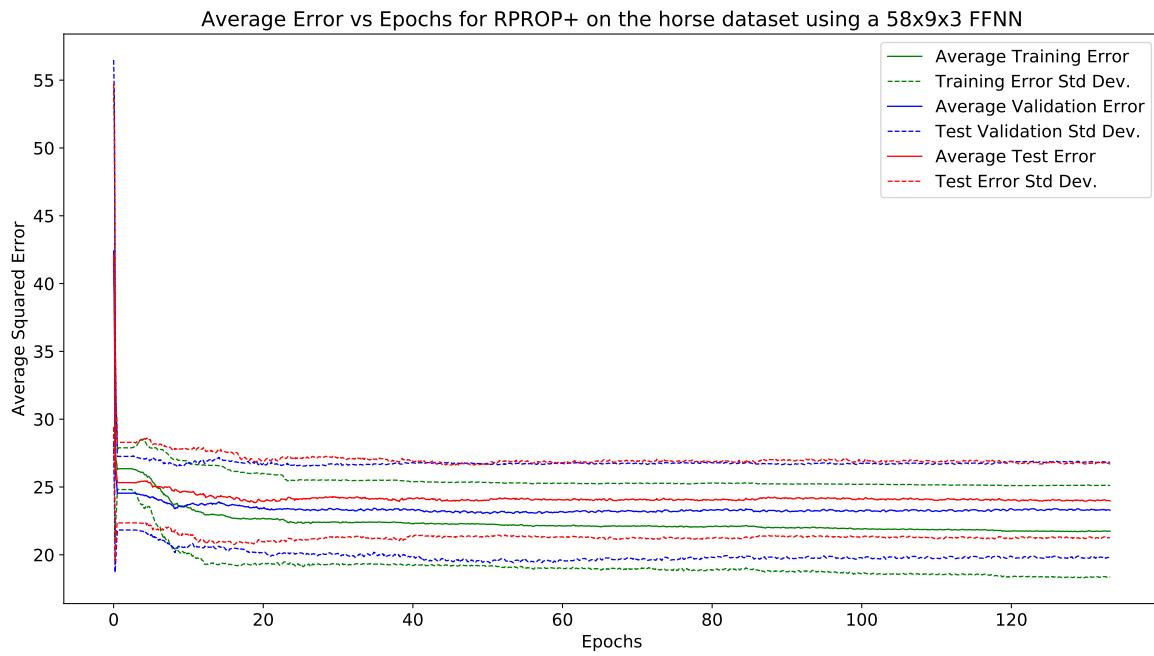
**Figure 433.** Graph of test error vs epochs for the gradient based algorithms

### 16.5.36 horse

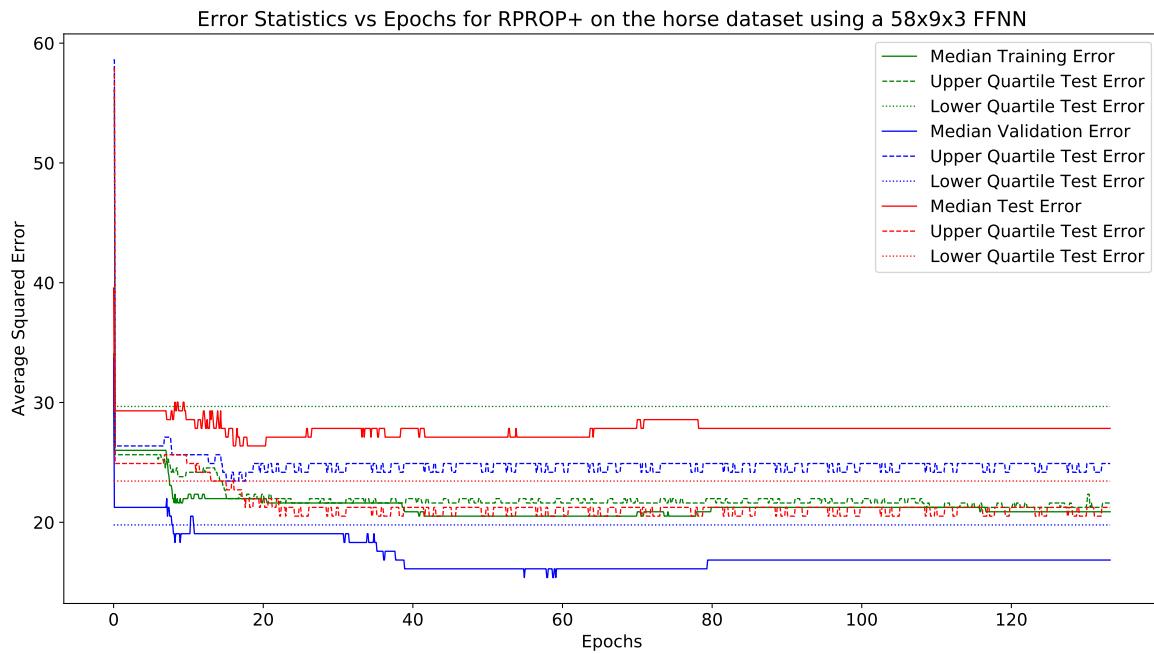
This section displays the results obtained using the horse algorithm.

#### 58 × 9 × 3 Architecture:

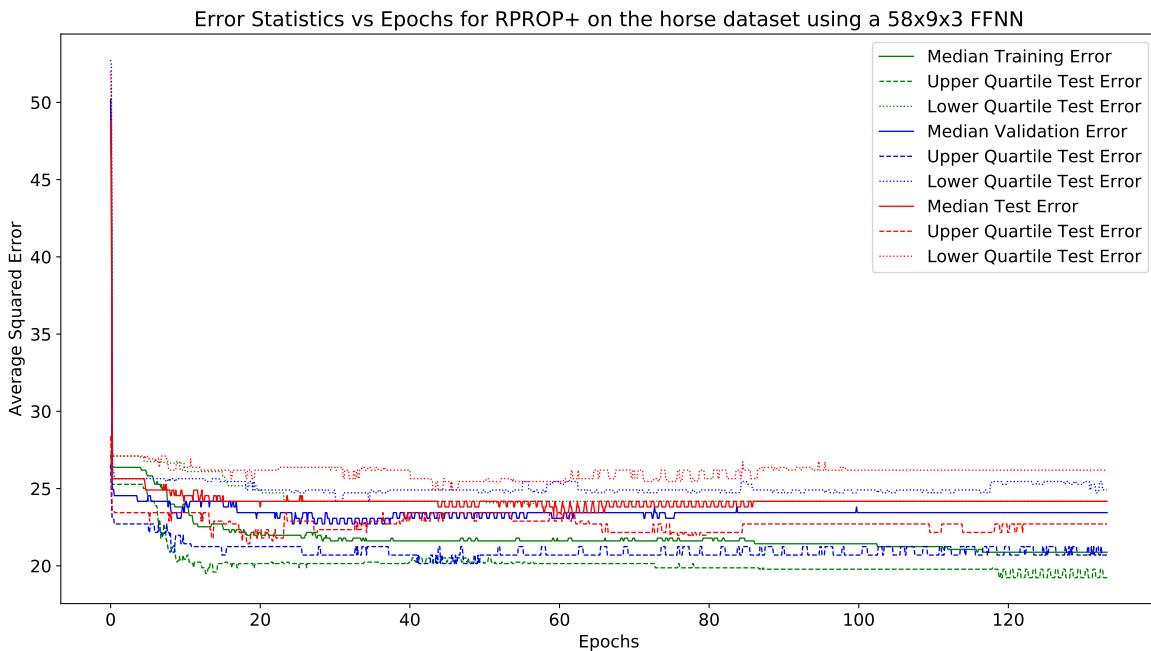
Fig. 434 shows the average and standard deviation of the test, training and validation errors. Fig. 435 shows the median and quartiles of the test, training and validation errors based on the best epoch. Finally, Fig. 436 shows the median and quartiles of the test, training and validation errors per mini-batch.



**Figure 434.** Graph of mean and standard deviation of errors vs epochs



**Figure 435.** Graph of test error vs epochs for the gradient based algorithms



**Figure 436.** Graph of test error vs epochs for the gradient based algorithms

## 16.6 heartc

This section displays the results obtained in the heartc dataset.

### 16.6.1 ADAGRAD

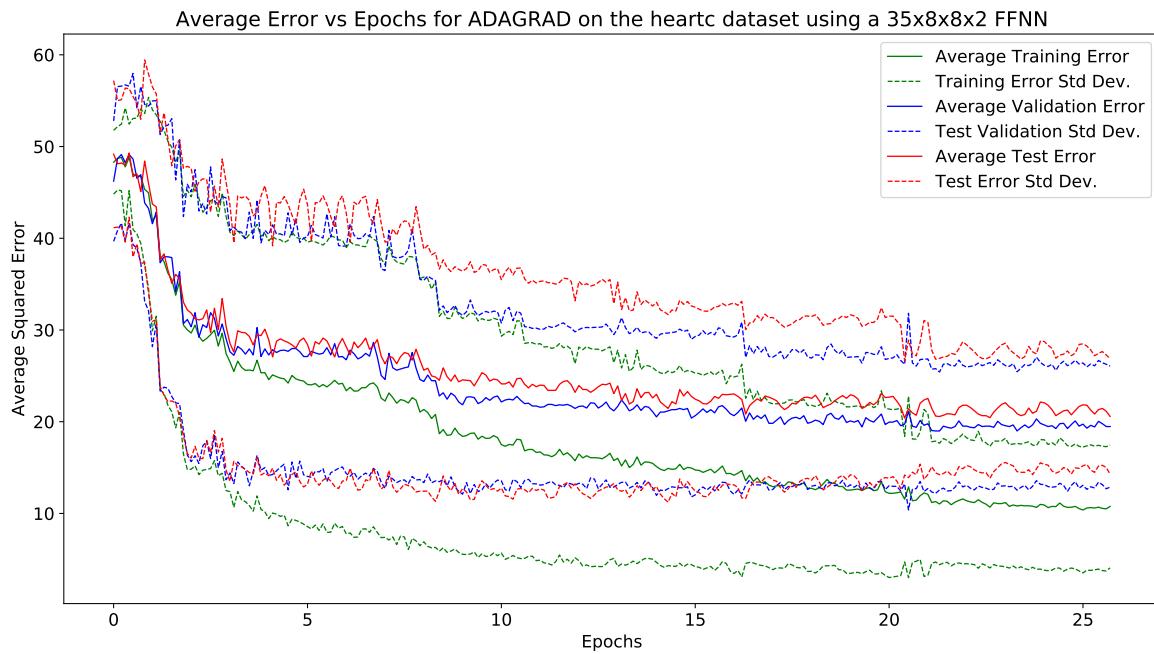
This section displays the results obtained using the ADAGRAD algorithm.

### 16.6.2 heartc

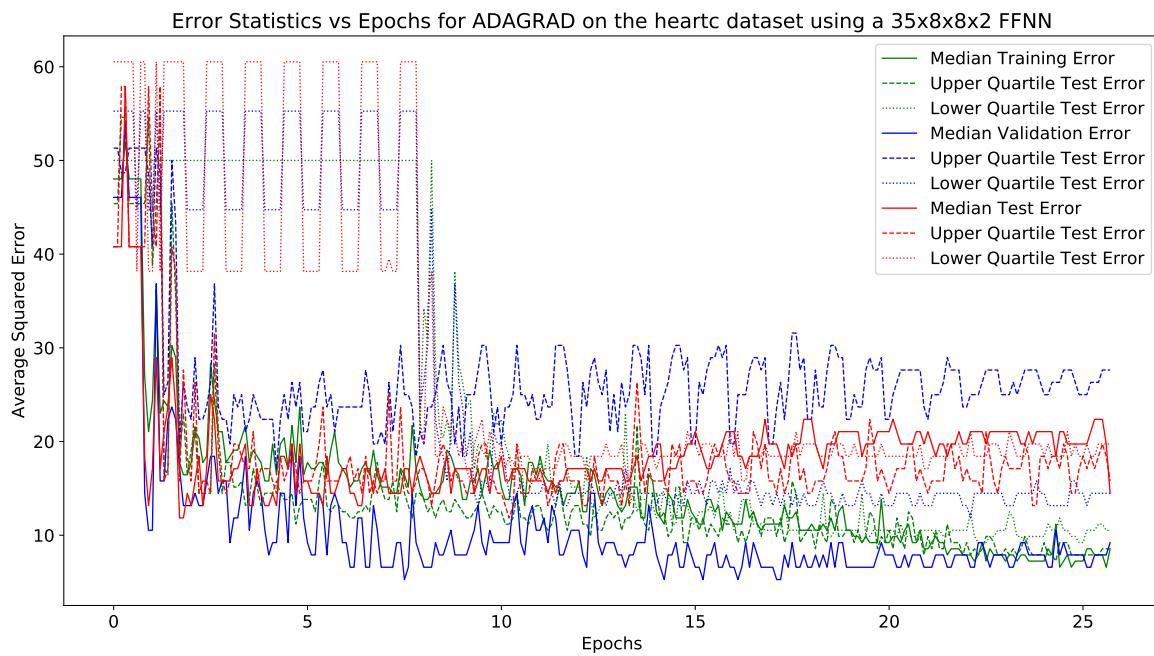
This section displays the results obtained using the heartc algorithm.

#### 35 × 8 × 8 × 2 Architecture:

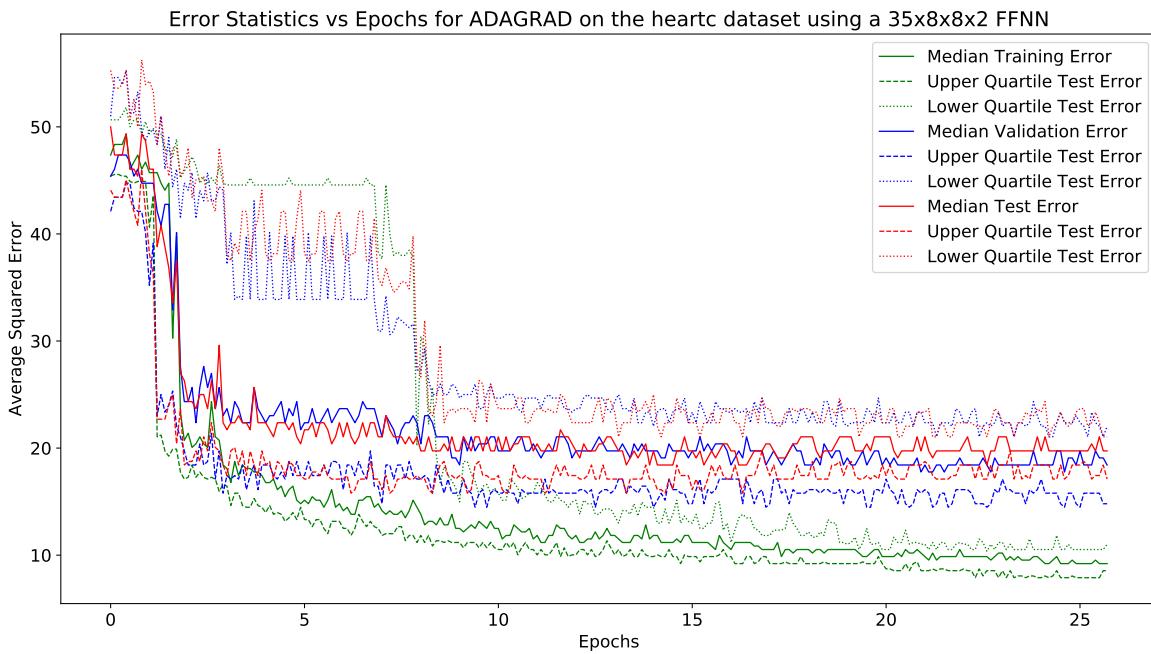
Fig. 437 shows the average and standard deviation of the test, training and validation errors. Fig. 438 shows the median and quartiles of the test, training and validation errors based on the best epoch. Finally, Fig. 439 shows the median and quartiles of the test, training and validation errors per mini-batch.



**Figure 437.** Graph of mean and standard deviation of errors vs epochs



**Figure 438.** Graph of test error vs epochs for the gradient based algorithms



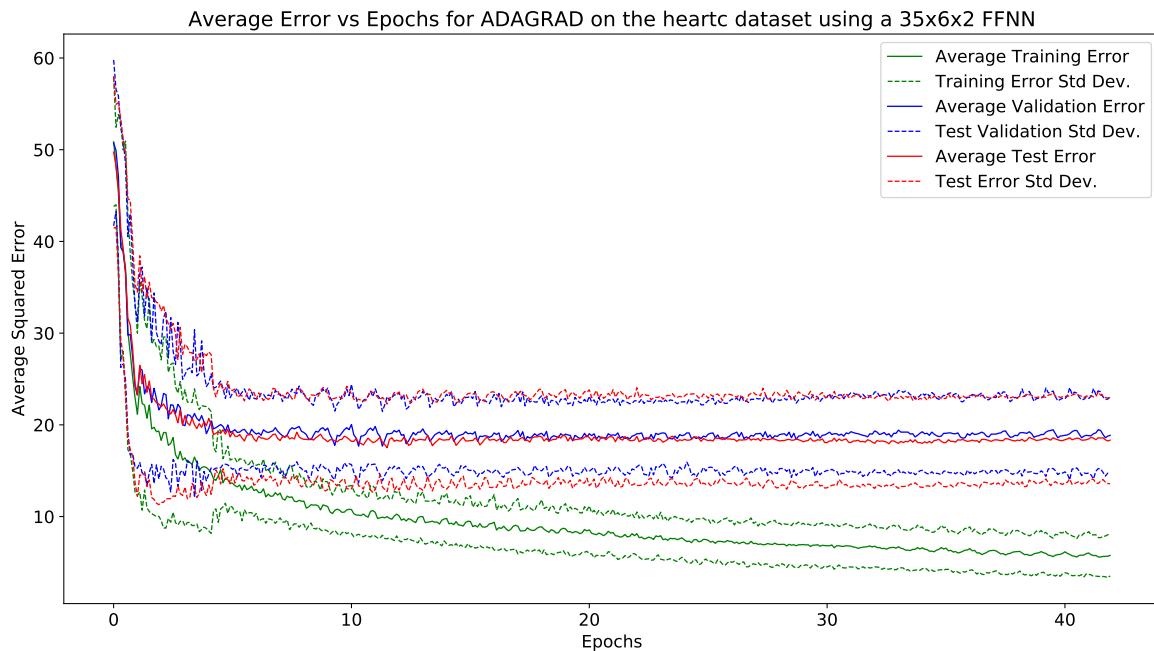
**Figure 439.** Graph of test error vs epochs for the gradient based algorithms

### 16.6.3 *heartc*

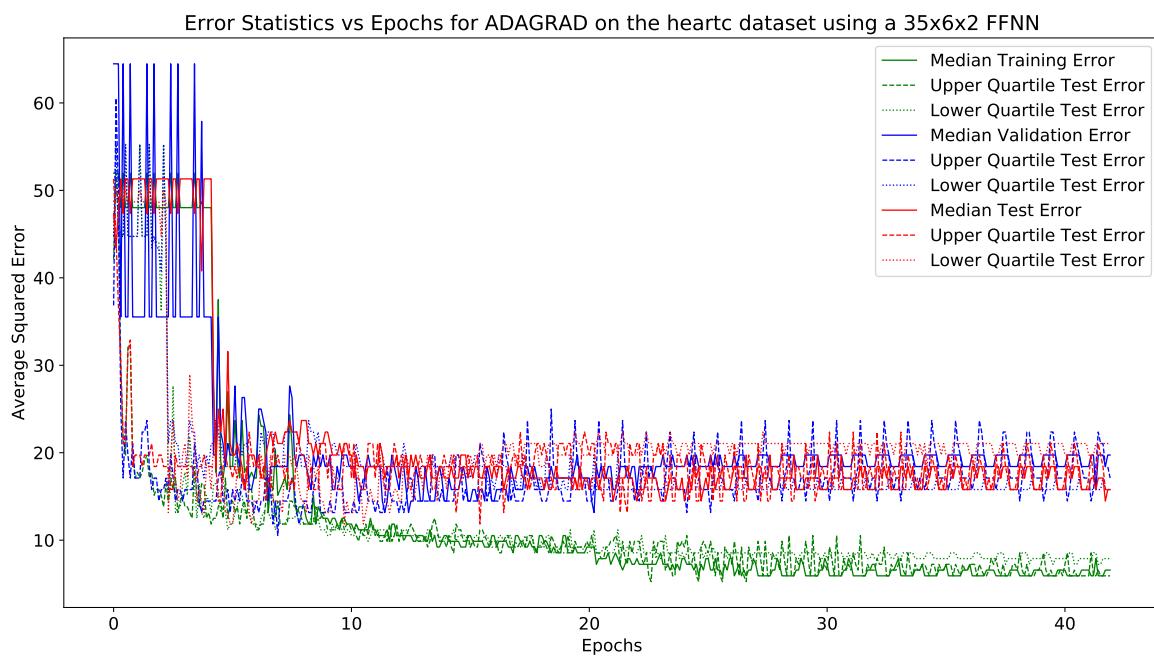
This section displays the results obtained using the *heartc* algorithm.

#### 35 × 6 × 2 Architecture:

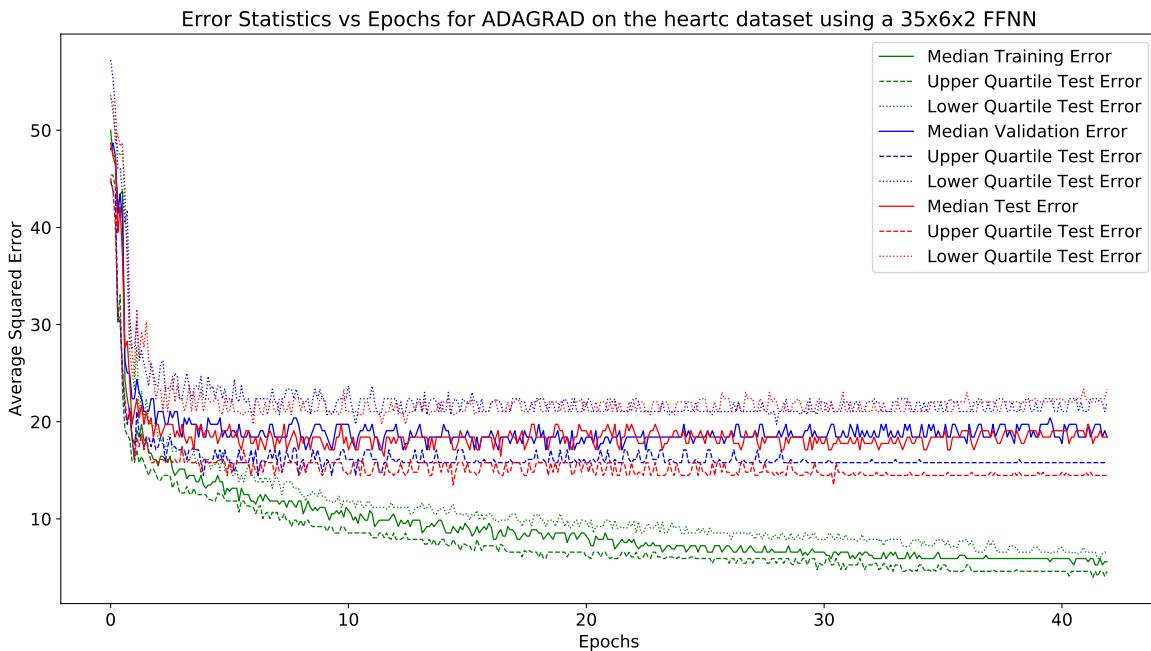
Fig. 440 shows the average and standard deviation of the test, training and validation errors. Fig. 441 shows the median and quartiles of the test, training and validation errors based on the best epoch. Finally, Fig. 442 shows the median and quartiles of the test, training and validation errors per mini-batch.



**Figure 440.** Graph of mean and standard deviation of errors vs epochs



**Figure 441.** Graph of test error vs epochs for the gradient based algorithms



**Figure 442.** Graph of test error vs epochs for the gradient based algorithms

#### 16.6.4 Backprop with Momentum

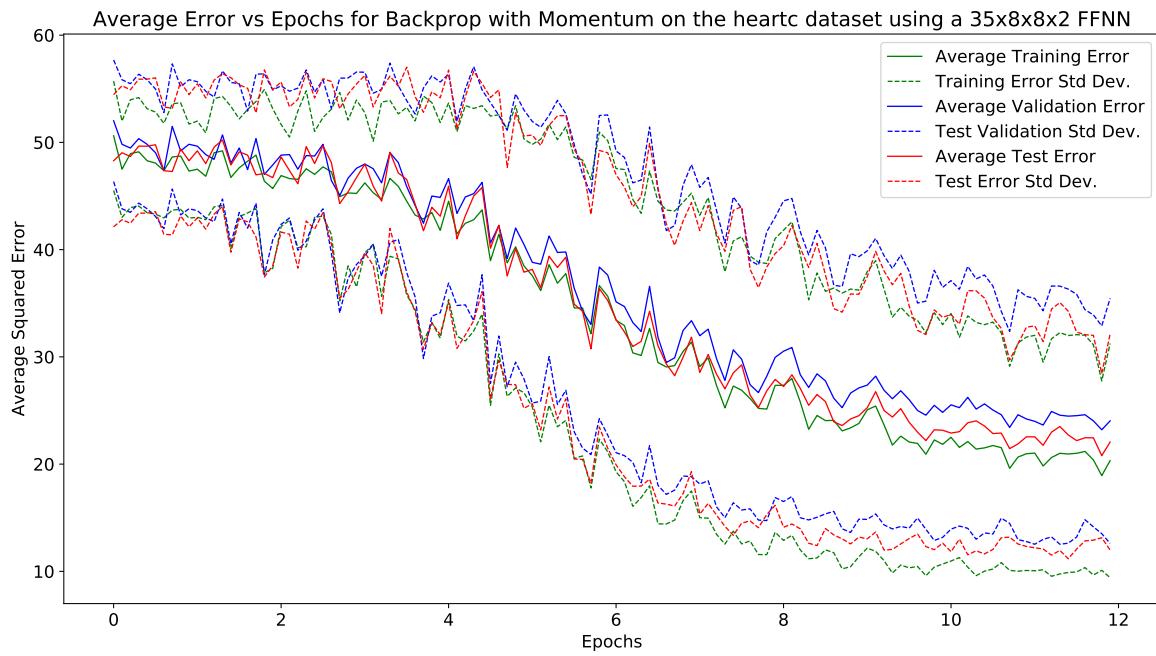
This section displays the results obtained using the Backprop with Momentum algorithm.

#### 16.6.5 heartc

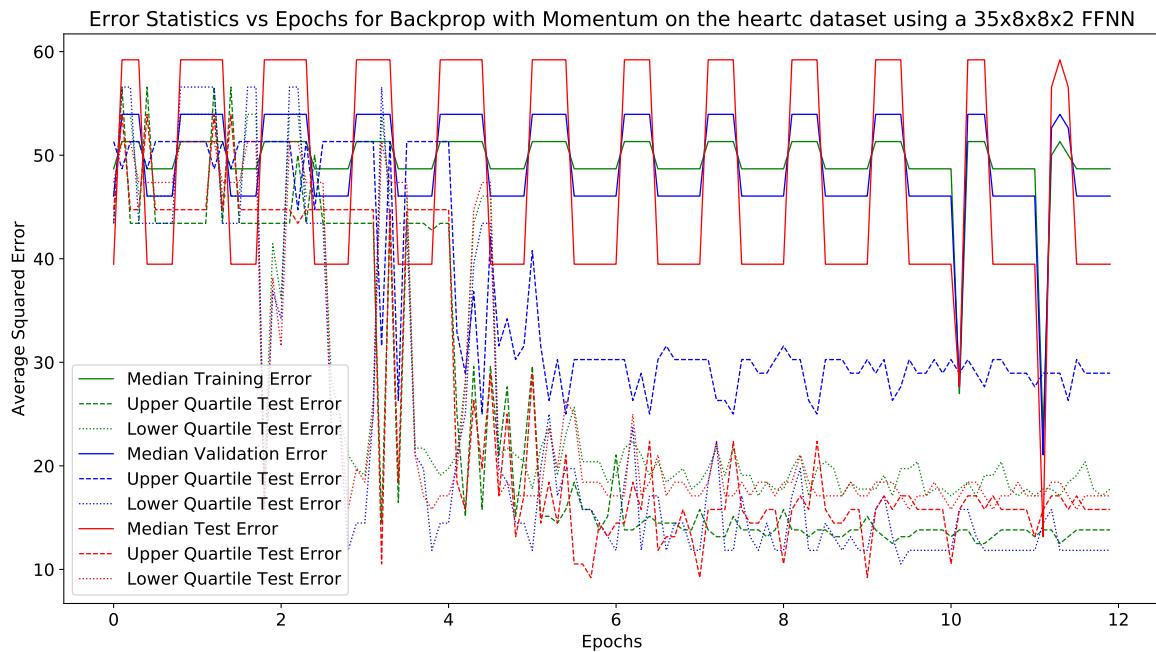
This section displays the results obtained using the heartc algorithm.

##### 35 × 8 × 8 × 2 Architecture:

Fig. 443 shows the average and standard deviation of the test, training and validation errors. Fig. 444 shows the median and quartiles of the test, training and validation errors based on the best epoch. Finally, Fig. 445 shows the median and quartiles of the test, training and validation errors per mini-batch.

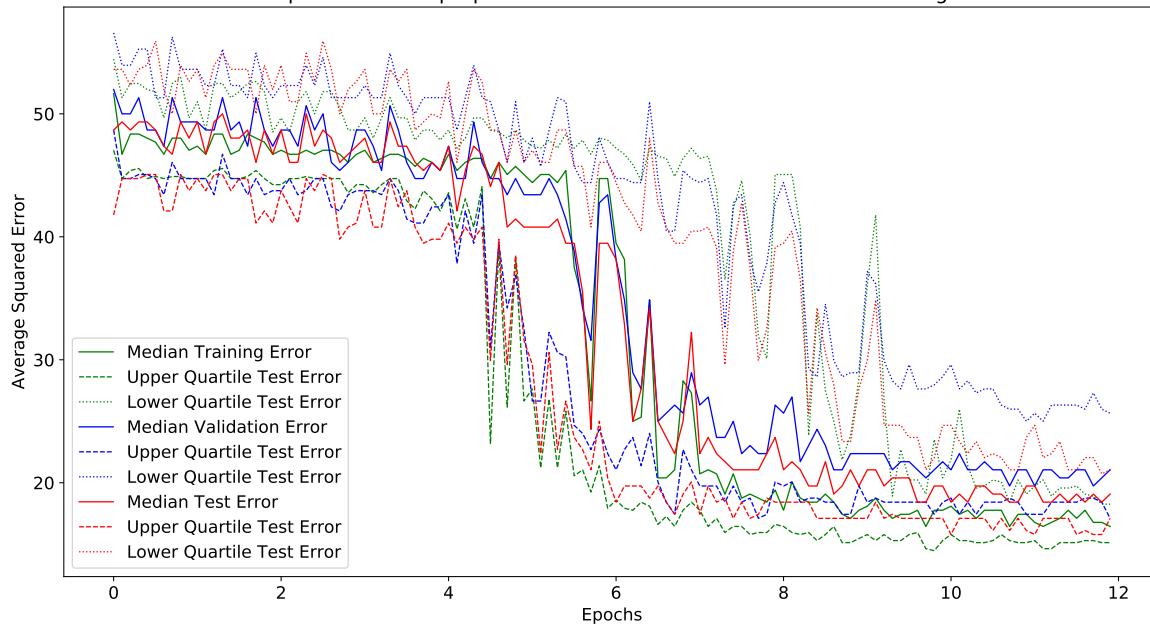


**Figure 443.** Graph of mean and standard deviation of errors vs epochs



**Figure 444.** Graph of test error vs epochs for the gradient based algorithms

Error Statistics vs Epochs for Backprop with Momentum on the heartc dataset using a 35x8x8x2 FFNN



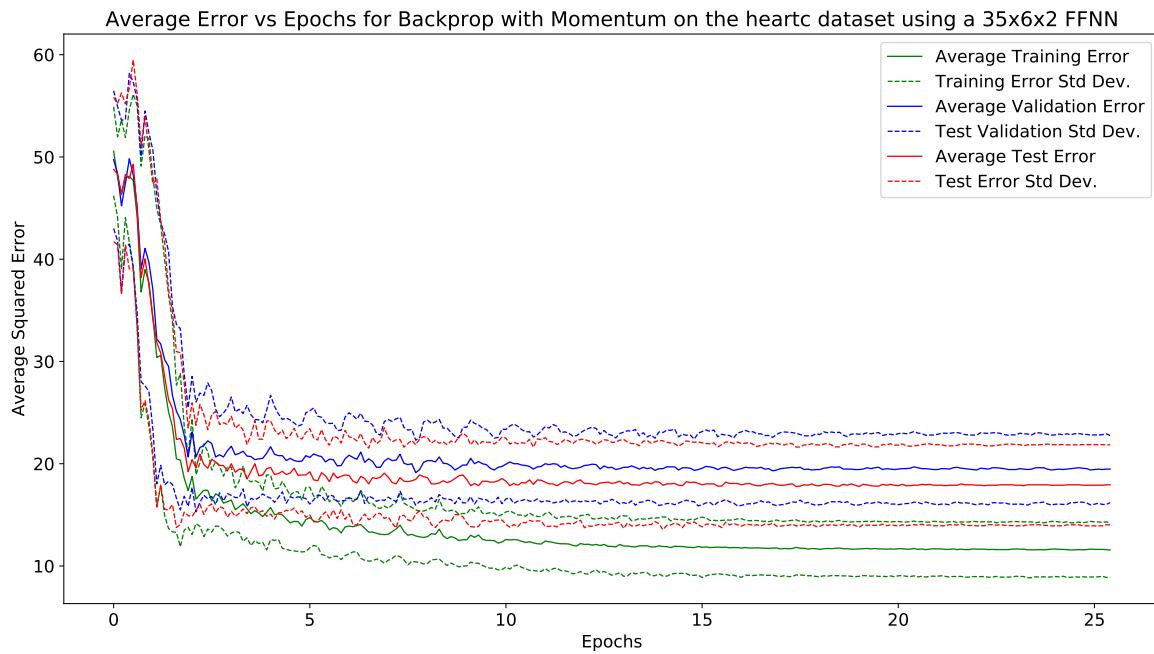
**Figure 445.** Graph of test error vs epochs for the gradient based algorithms

### 16.6.6 *heartc*

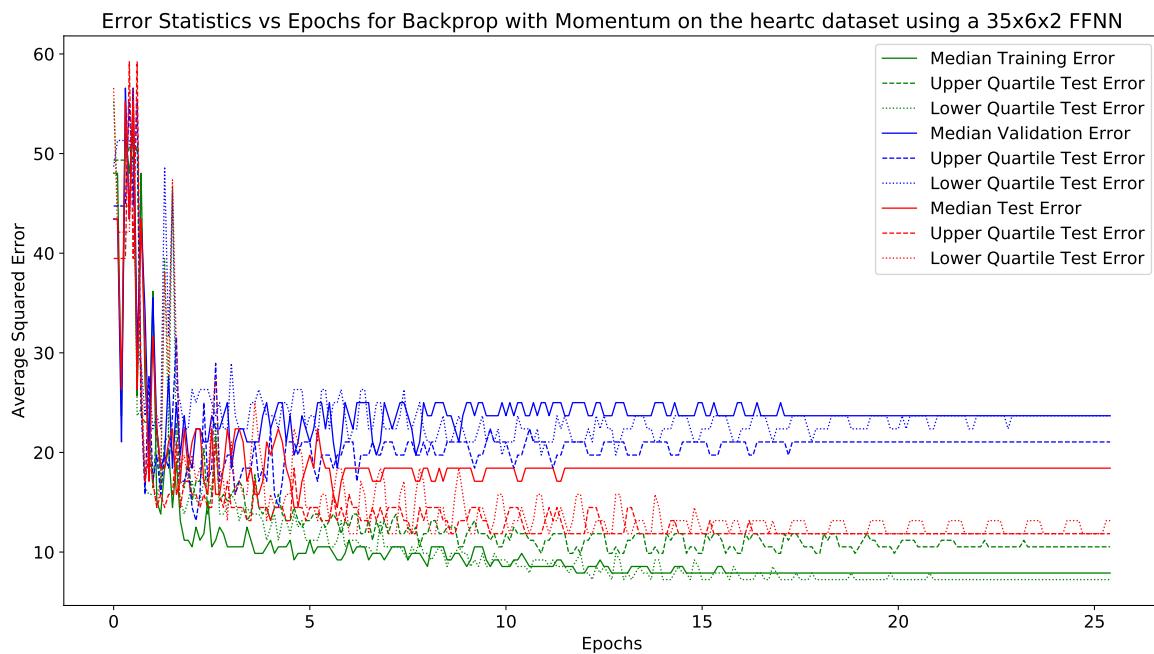
This section displays the results obtained using the *heartc* algorithm.

#### 35 × 6 × 2 Architecture:

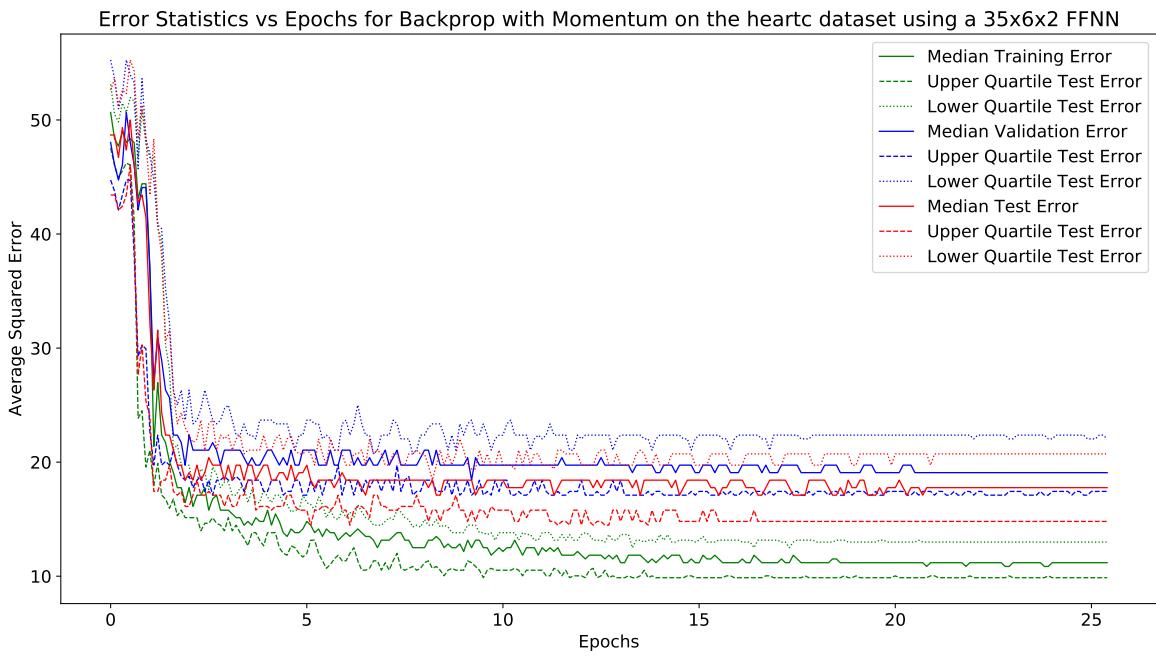
Fig. 446 shows the average and standard deviation of the test, training and validation errors. Fig. 447 shows the median and quartiles of the test, training and validation errors based on the best epoch. Finally, Fig. 448 shows the median and quartiles of the test, training and validation errors per mini-batch.



**Figure 446.** Graph of mean and standard deviation of errors vs epochs



**Figure 447.** Graph of test error vs epochs for the gradient based algorithms



**Figure 448.** Graph of test error vs epochs for the gradient based algorithms

### 16.6.7 back-propagation

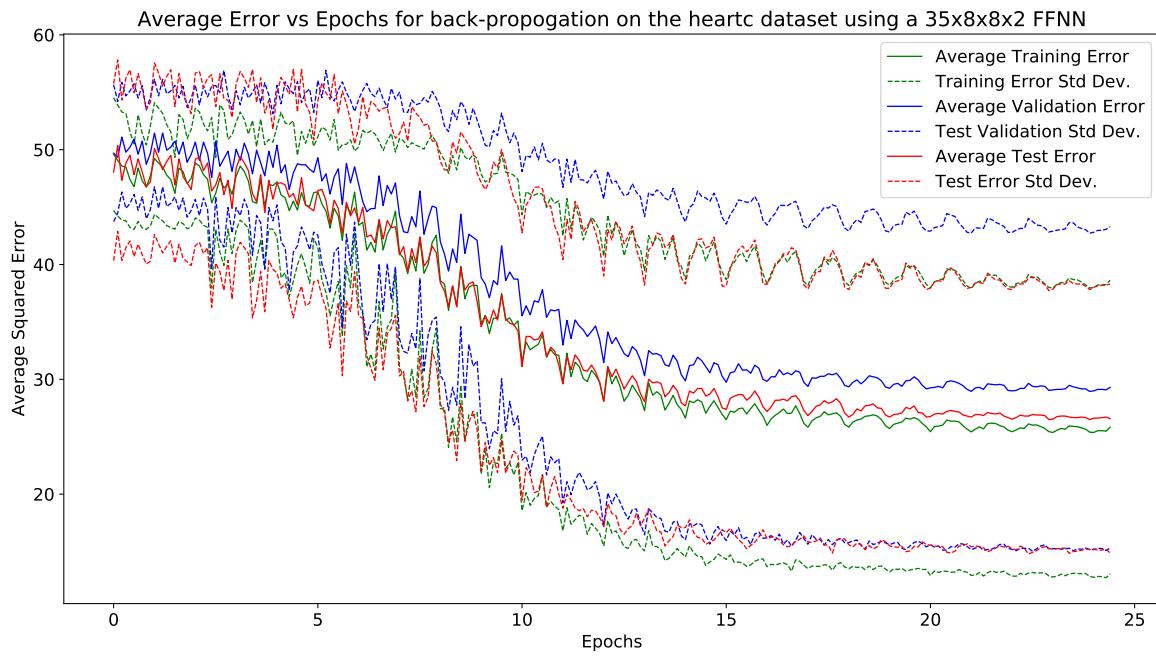
This section displays the results obtained using the back-propagation algorithm.

### 16.6.8 heartc

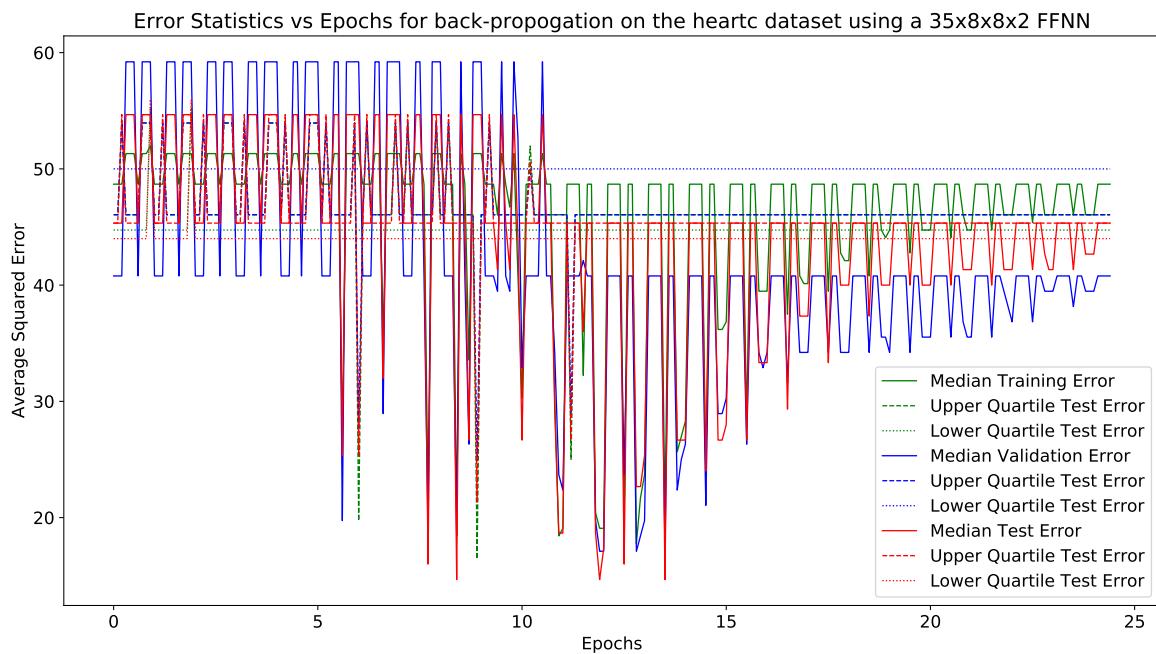
This section displays the results obtained using the heartc algorithm.

#### 35 × 8 × 8 × 2 Architecture:

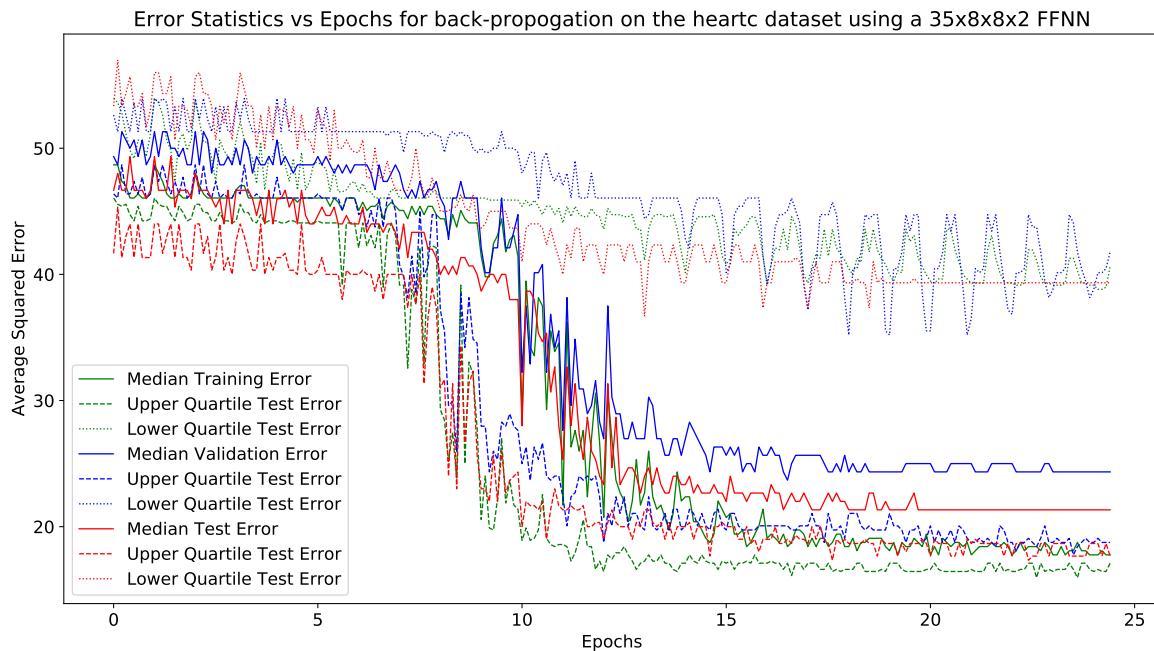
Fig. 449 shows the average and standard deviation of the test, training and validation errors. Fig. 450 shows the median and quartiles of the test, training and validation errors based on the best epoch. Finally, Fig. 451 shows the median and quartiles of the test, training and validation errors per mini-batch.



**Figure 449.** Graph of mean and standard deviation of errors vs epochs



**Figure 450.** Graph of test error vs epochs for the gradient based algorithms



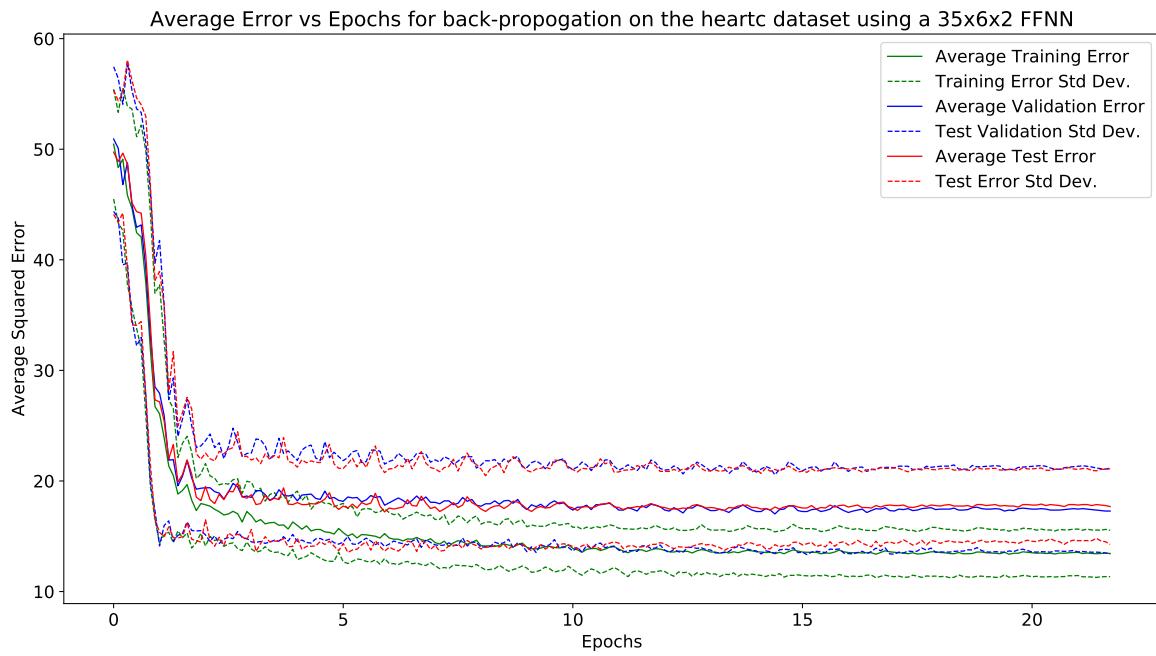
**Figure 451.** Graph of test error vs epochs for the gradient based algorithms

### 16.6.9 *heartc*

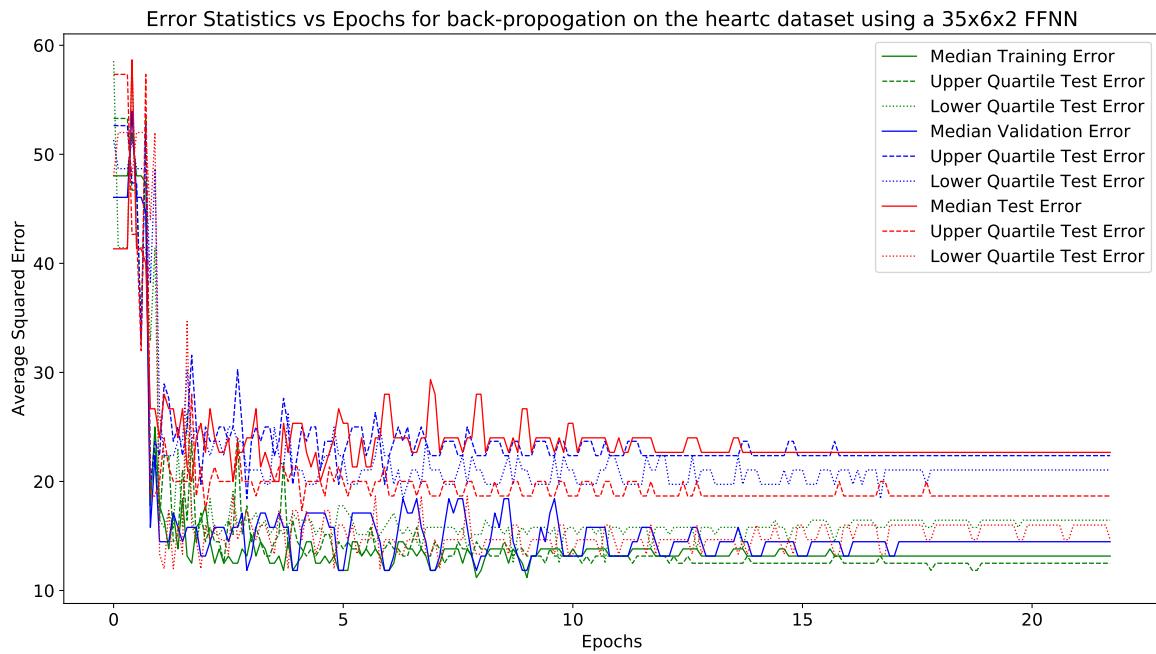
This section displays the results obtained using the *heartc* algorithm.

#### 35 × 6 × 2 Architecture:

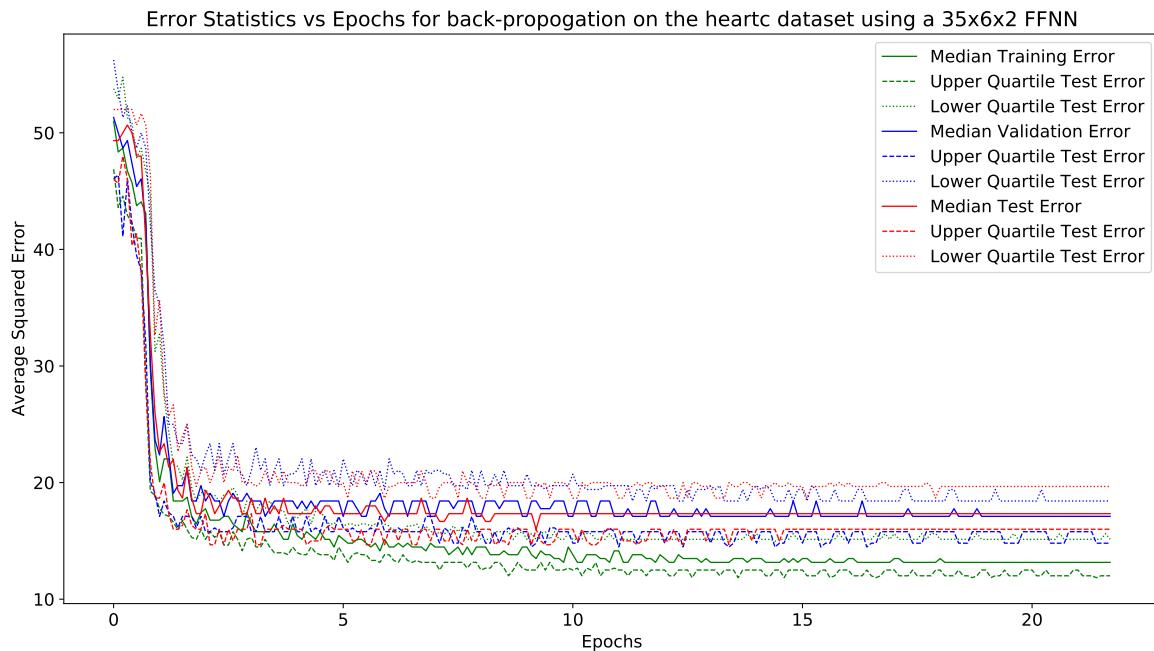
Fig. 452 shows the average and standard deviation of the test, training and validation errors. Fig. 453 shows the median and quartiles of the test, training and validation errors based on the best epoch. Finally, Fig. 454 shows the median and quartiles of the test, training and validation errors per mini-batch.



**Figure 452.** Graph of mean and standard deviation of errors vs epochs



**Figure 453.** Graph of test error vs epochs for the gradient based algorithms



**Figure 454.** Graph of test error vs epochs for the gradient based algorithms

### 16.6.10 GA1

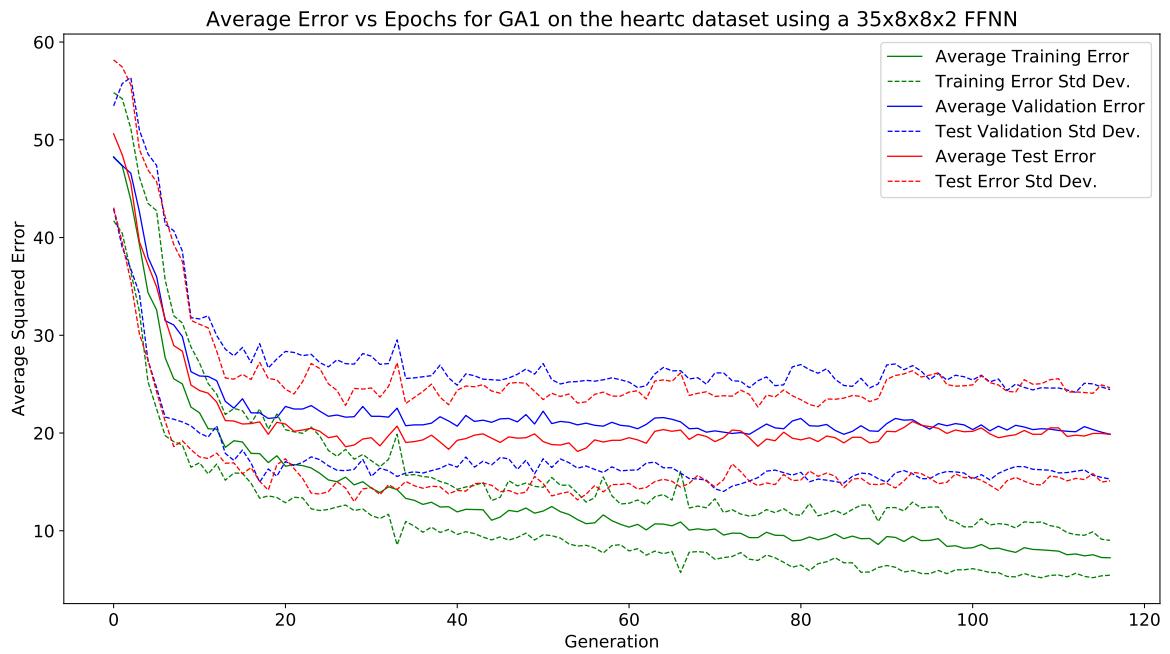
This section displays the results obtained using the GA1 algorithm.

### 16.6.11 heartc

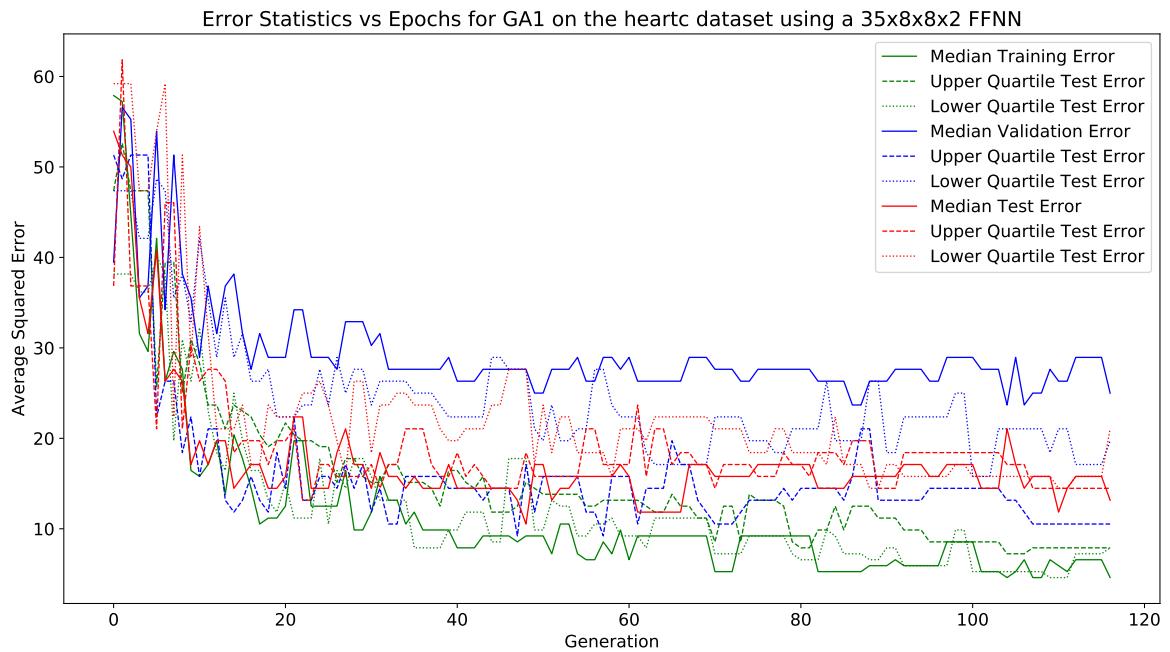
This section displays the results obtained using the heartc algorithm.

#### 35 × 8 × 8 × 2 Architecture:

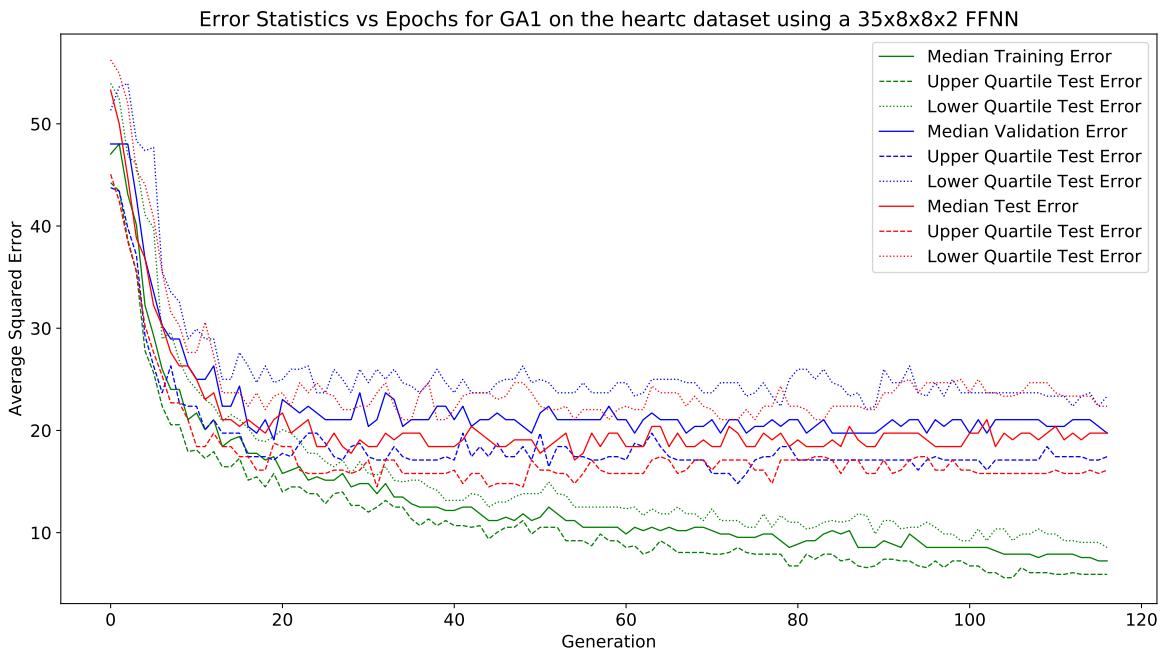
Fig. 455 shows the average and standard deviation of the test, training and validation errors. Fig. 456 shows the median and quartiles of the test, training and validation errors based on the best epoch. Finally, Fig. 457 shows the median and quartiles of the test, training and validation errors per mini-batch.



**Figure 455.** Graph of mean and standard deviation of errors vs epochs



**Figure 456.** Graph of test error vs epochs for the gradient based algorithms



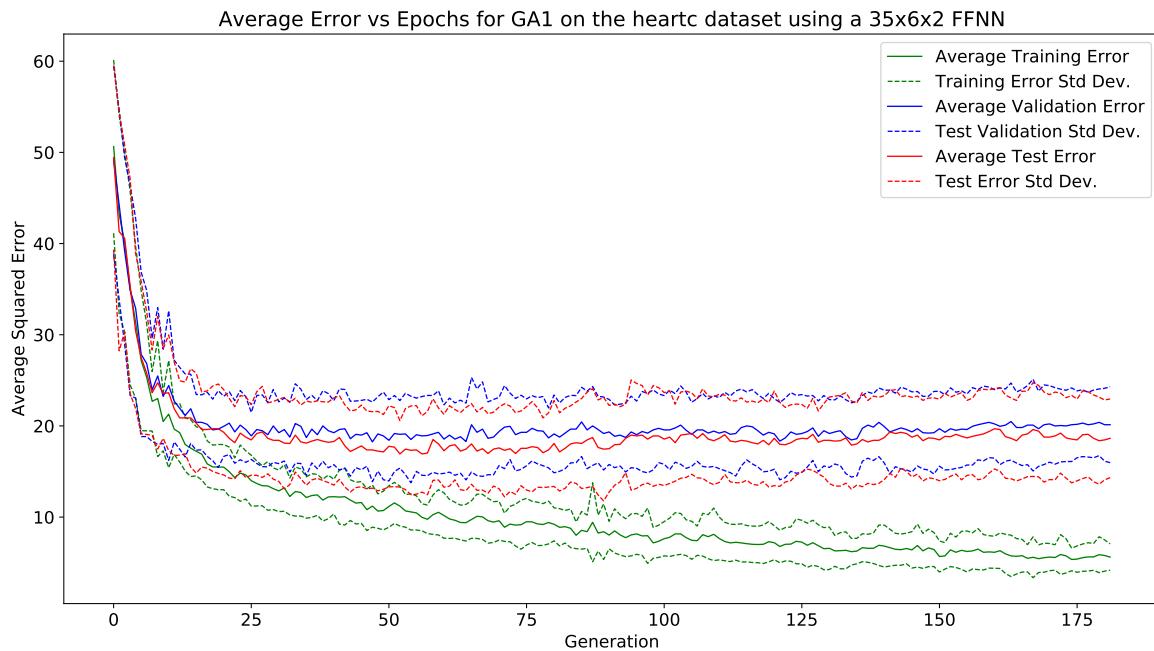
**Figure 457.** Graph of test error vs epochs for the gradient based algorithms

### 16.6.12 heartc

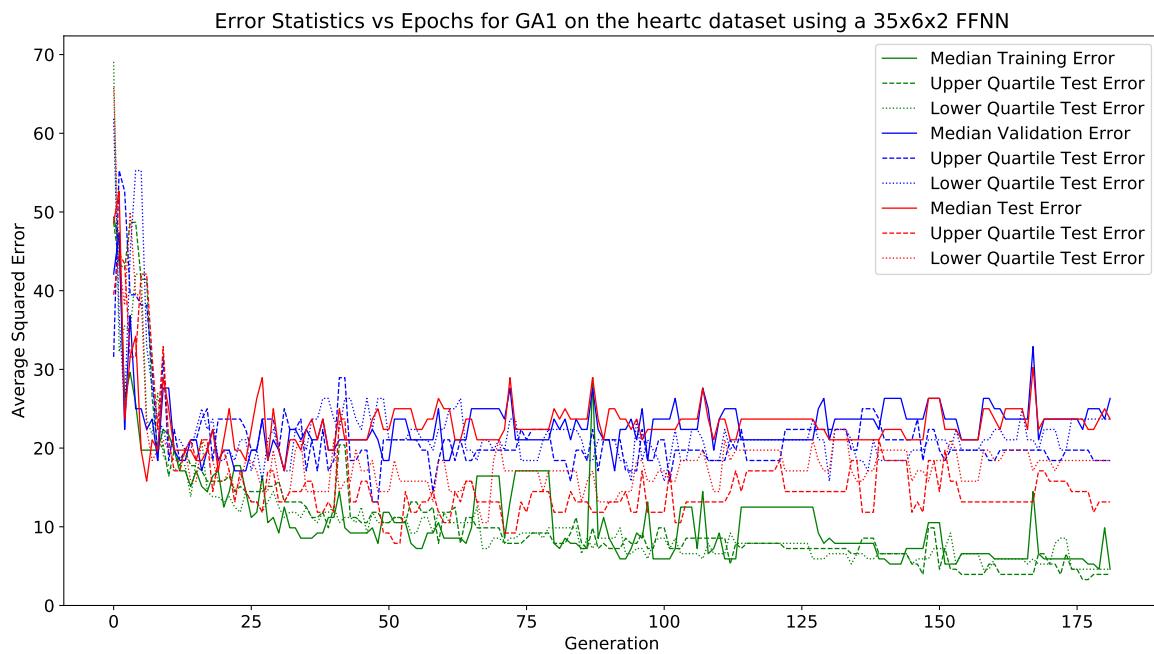
This section displays the results obtained using the heartc algorithm.

#### 35 × 6 × 2 Architecture:

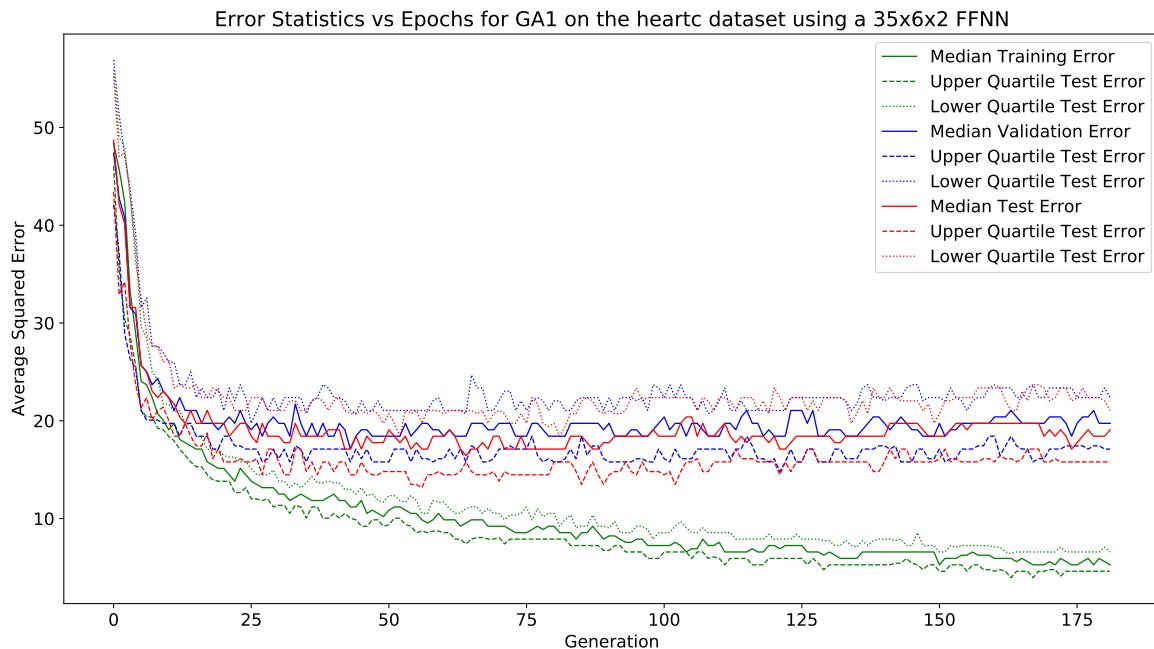
Fig. 458 shows the average and standard deviation of the test, training and validation errors. Fig. 459 shows the median and quartiles of the test, training and validation errors based on the best epoch. Finally, Fig. 460 shows the median and quartiles of the test, training and validation errors per mini-batch.



**Figure 458.** Graph of mean and standard deviation of errors vs epochs



**Figure 459.** Graph of test error vs epochs for the gradient based algorithms



**Figure 460.** Graph of test error vs epochs for the gradient based algorithms

### 16.6.13 GA2

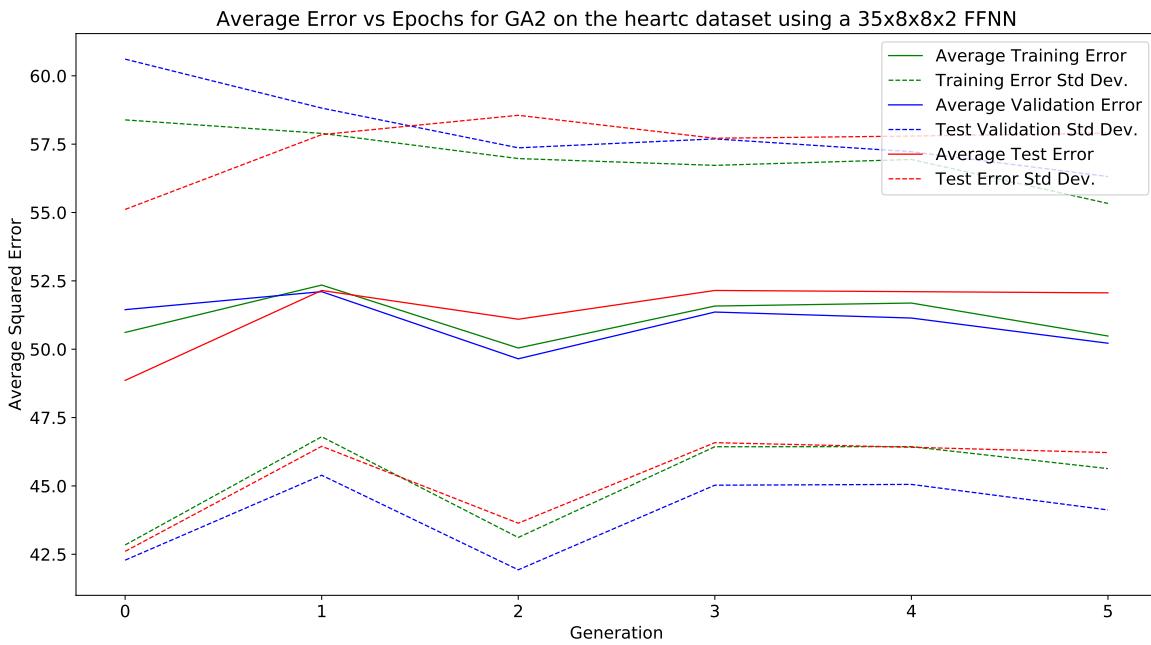
This section displays the results obtained using the GA2 algorithm.

### 16.6.14 heartc

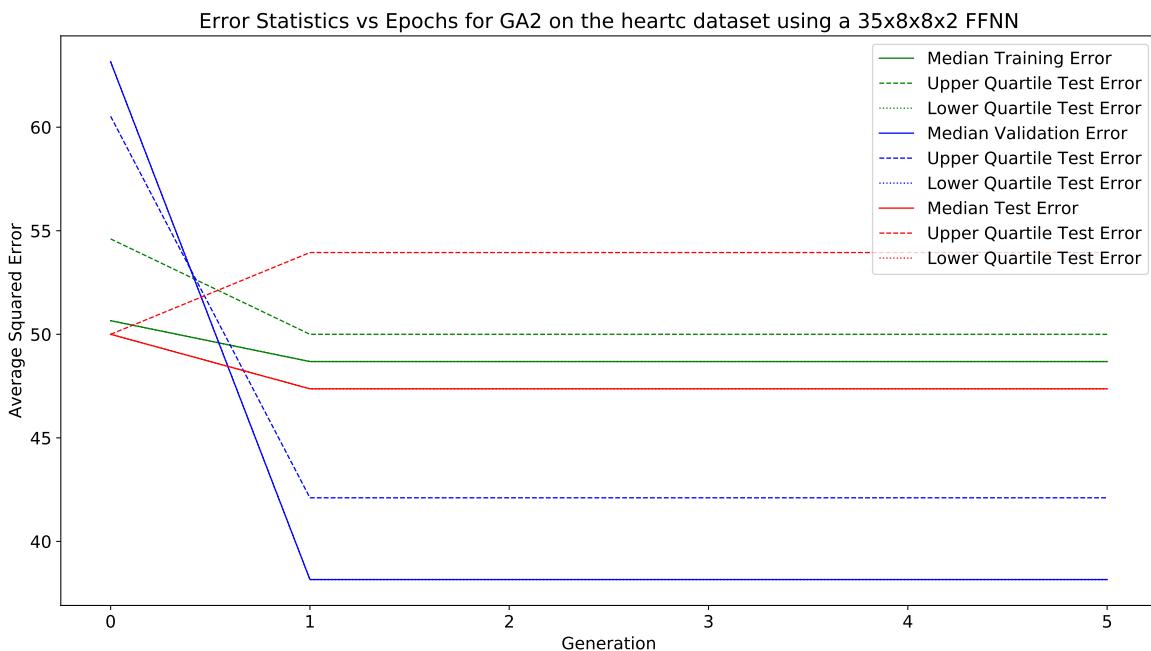
This section displays the results obtained using the heartc algorithm.

#### 35 × 8 × 8 × 2 Architecture:

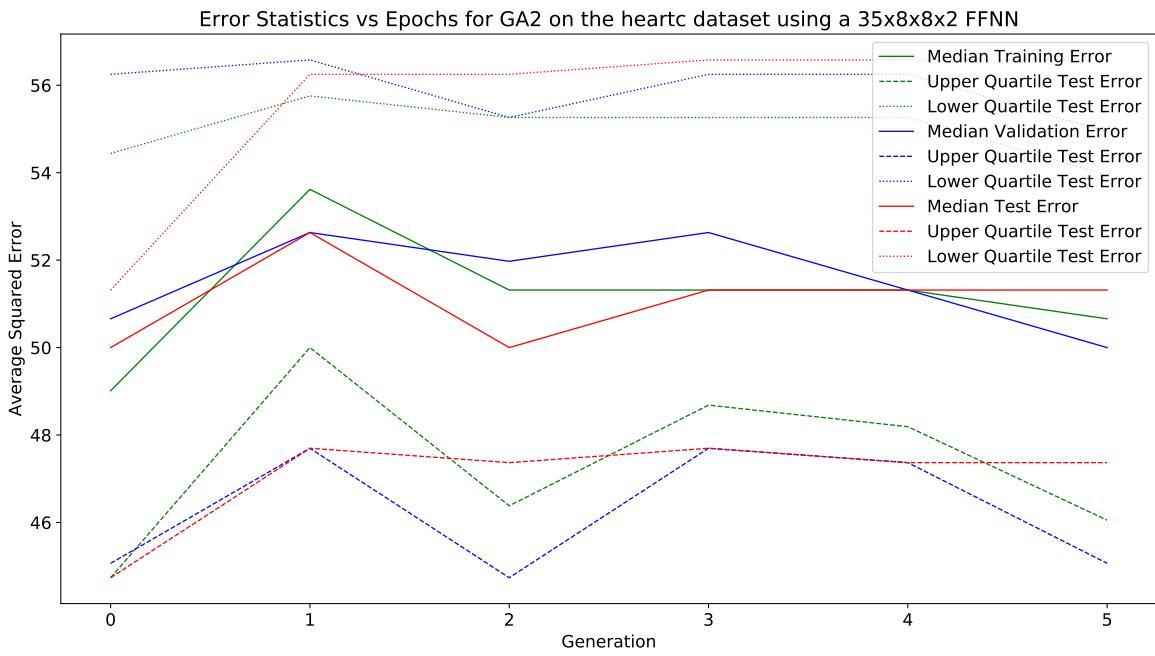
Fig. 461 shows the average and standard deviation of the test, training and validation errors. Fig. 462 shows the median and quartiles of the test, training and validation errors based on the best epoch. Finally, Fig. 463 shows the median and quartiles of the test, training and validation errors per mini-batch.



**Figure 461.** Graph of mean and standard deviation of errors vs epochs



**Figure 462.** Graph of test error vs epochs for the gradient based algorithms



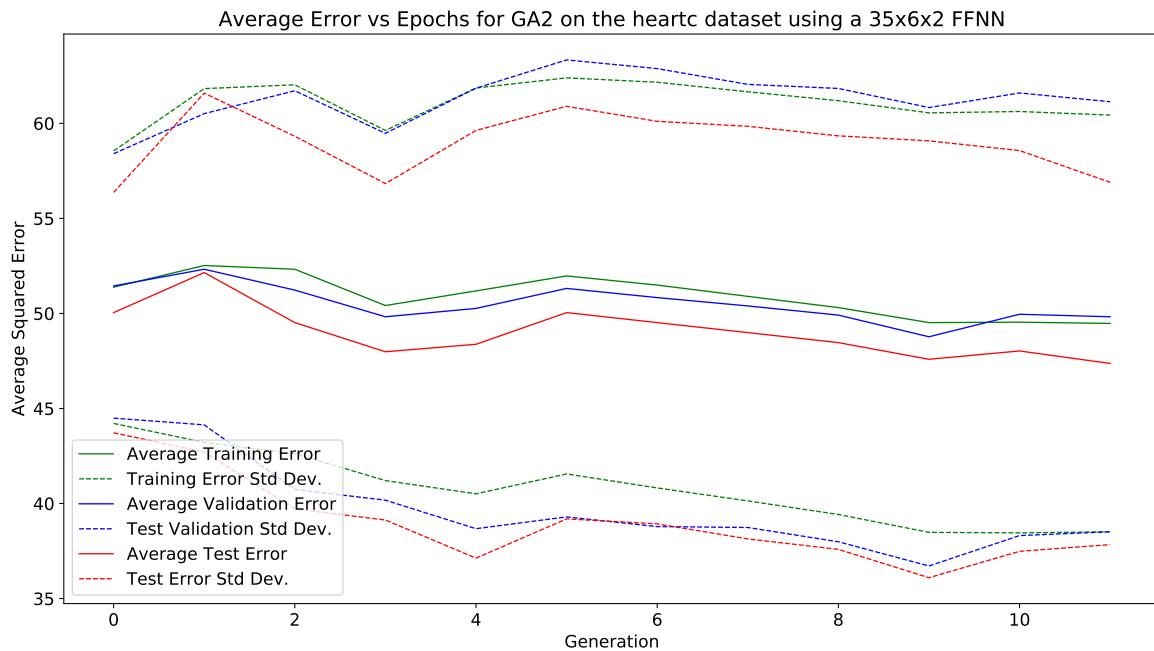
**Figure 463.** Graph of test error vs epochs for the gradient based algorithms

### 16.6.15 heartc

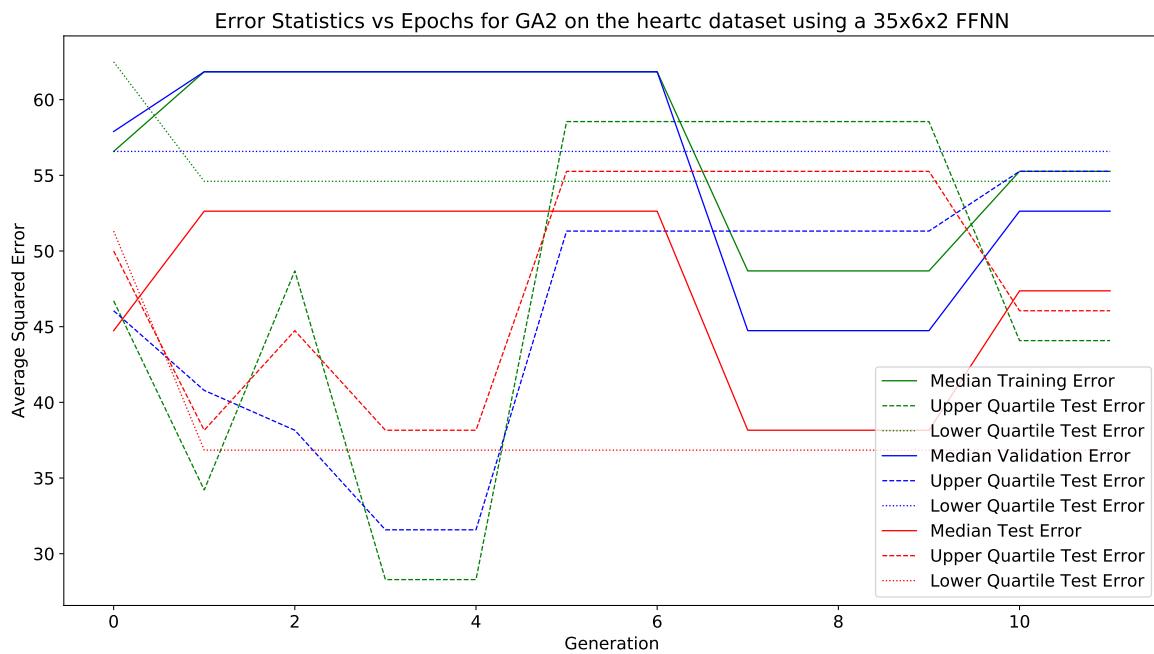
This section displays the results obtained using the heartc algorithm.

#### 35 × 6 × 2 Architecture:

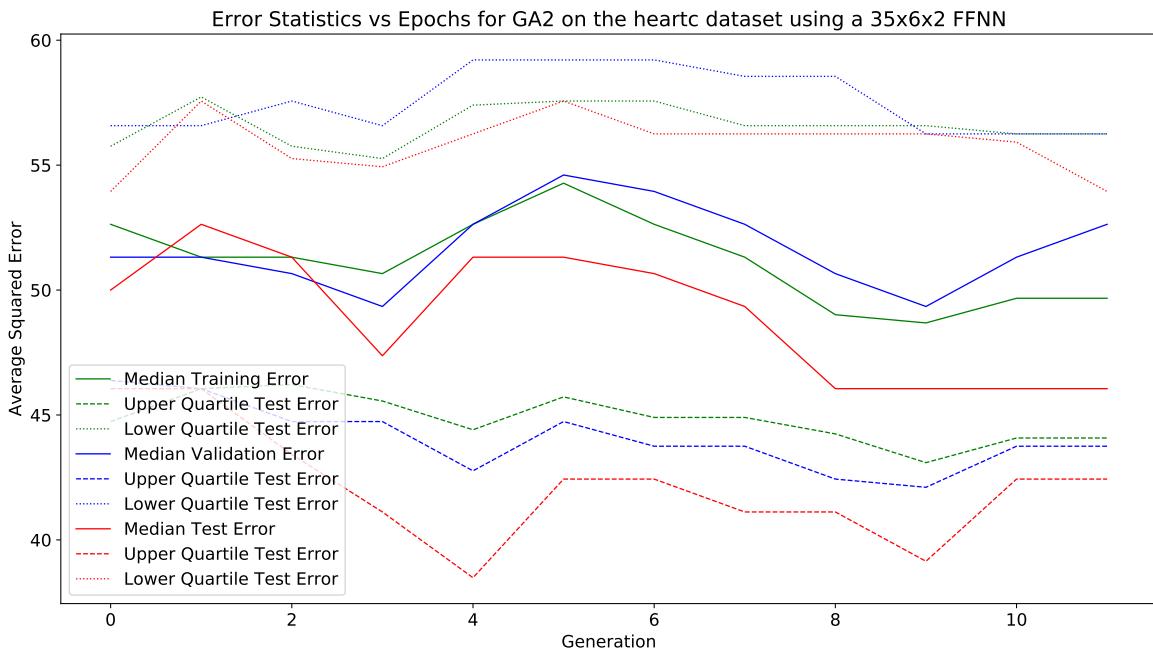
Fig. 464 shows the average and standard deviation of the test, training and validation errors. Fig. 465 shows the median and quartiles of the test, training and validation errors based on the best epoch. Finally, Fig. 466 shows the median and quartiles of the test, training and validation errors per mini-batch.



**Figure 464.** Graph of mean and standard deviation of errors vs epochs



**Figure 465.** Graph of test error vs epochs for the gradient based algorithms



**Figure 466.** Graph of test error vs epochs for the gradient based algorithms

### 16.6.16 GA3

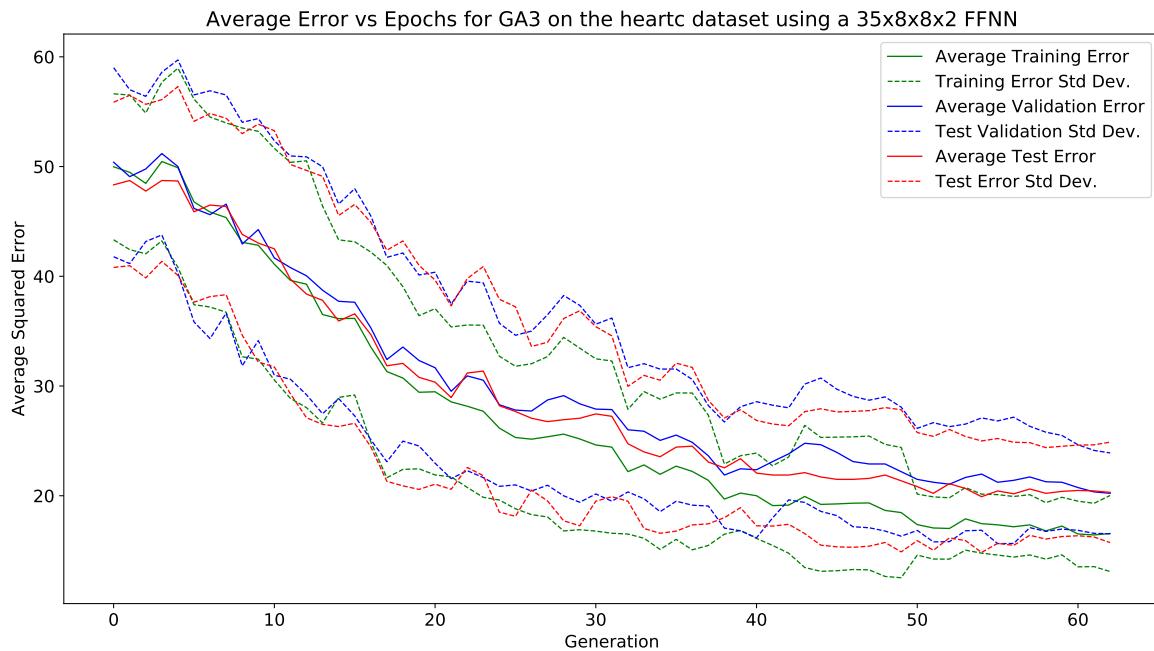
This section displays the results obtained using the GA3 algorithm.

### 16.6.17 heartc

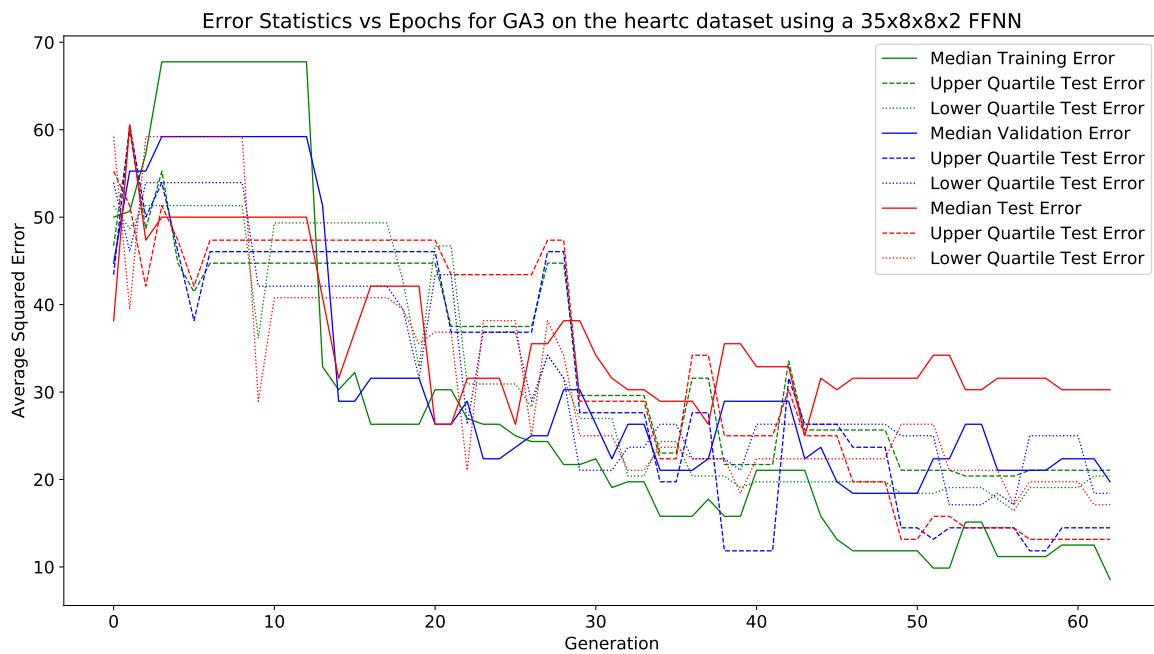
This section displays the results obtained using the heartc algorithm.

#### 35 × 8 × 8 × 2 Architecture:

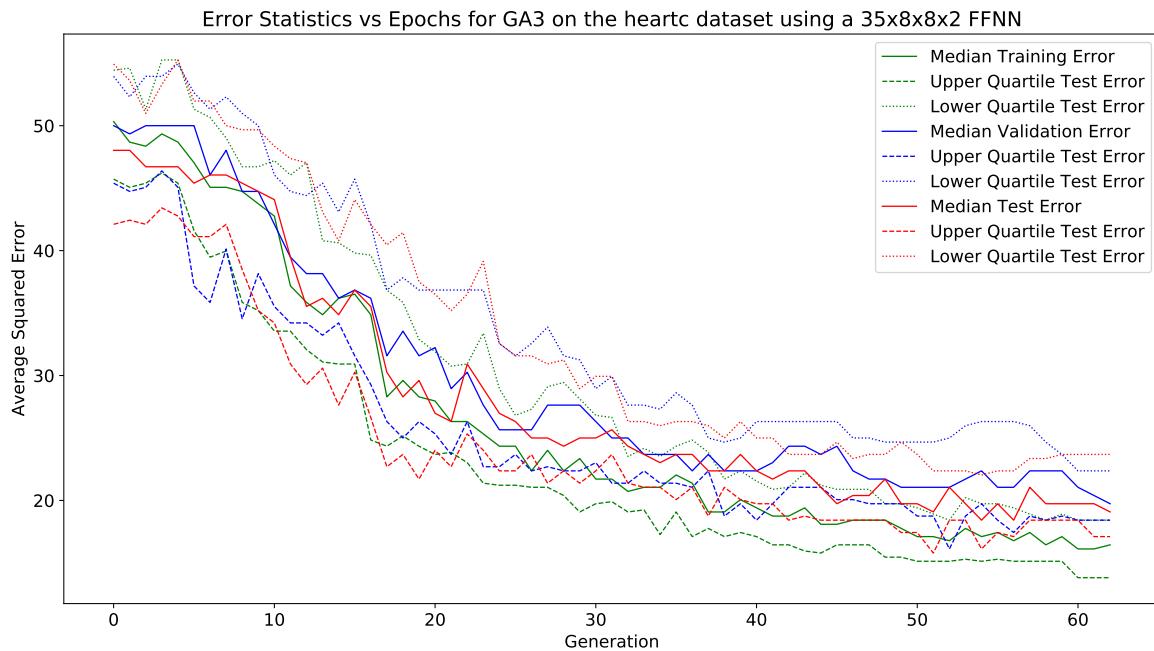
Fig. 467 shows the average and standard deviation of the test, training and validation errors. Fig. 468 shows the median and quartiles of the test, training and validation errors based on the best epoch. Finally, Fig. 469 shows the median and quartiles of the test, training and validation errors per mini-batch.



**Figure 467.** Graph of mean and standard deviation of errors vs epochs



**Figure 468.** Graph of test error vs epochs for the gradient based algorithms



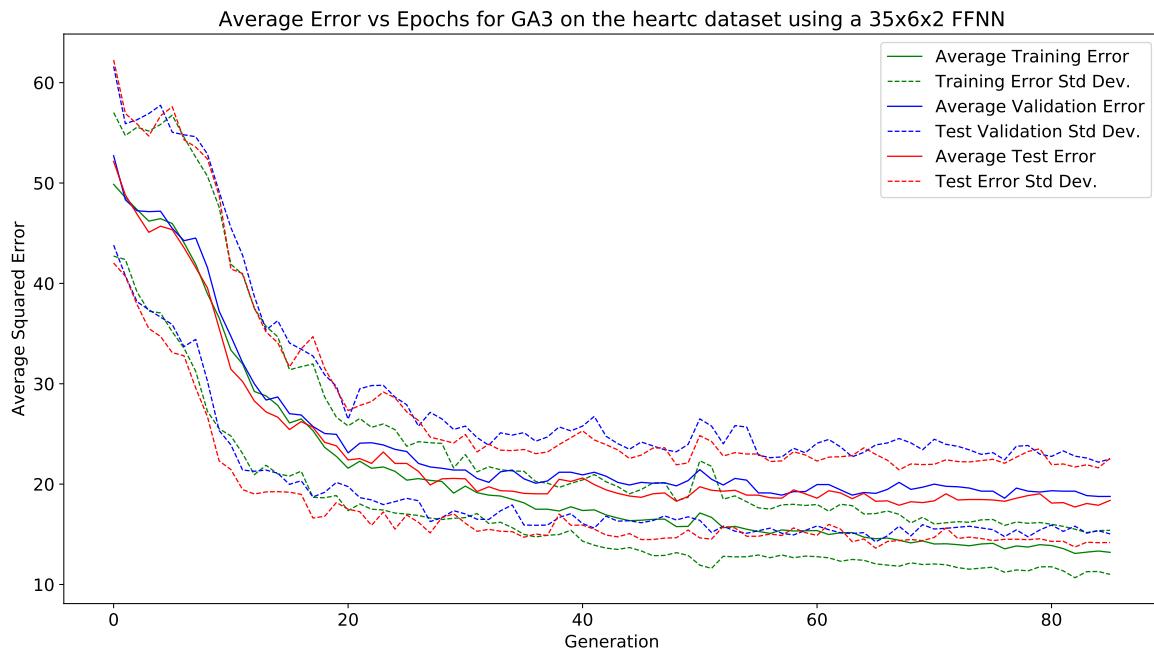
**Figure 469.** Graph of test error vs epochs for the gradient based algorithms

### 16.6.18 heartc

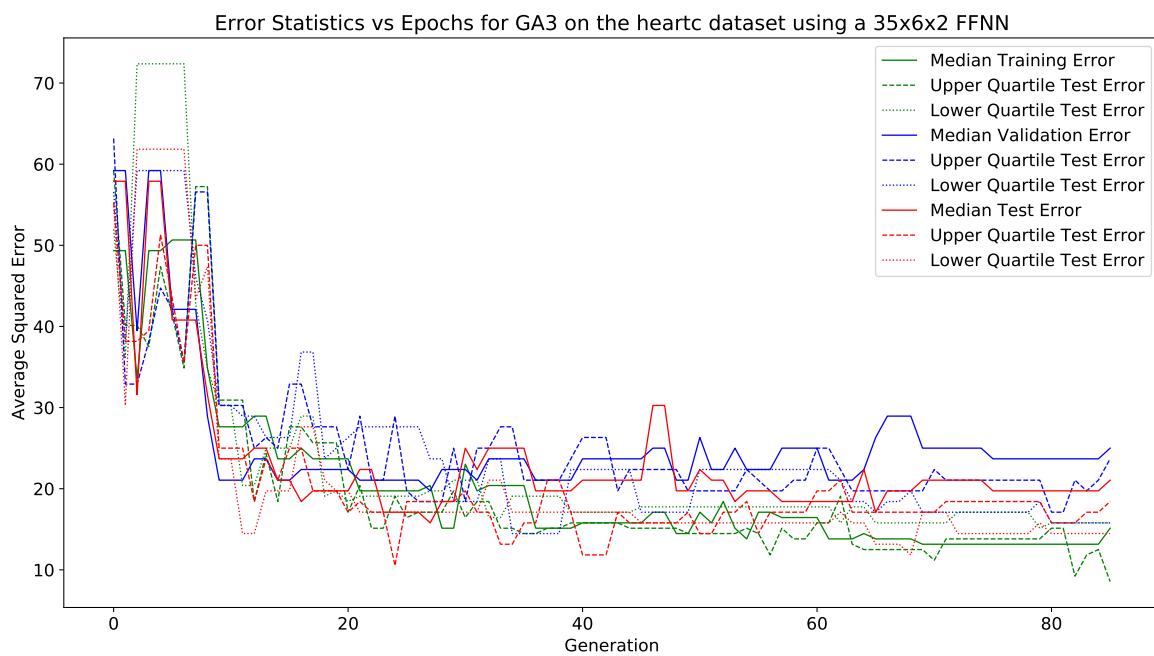
This section displays the results obtained using the heartc algorithm.

#### 35 × 6 × 2 Architecture:

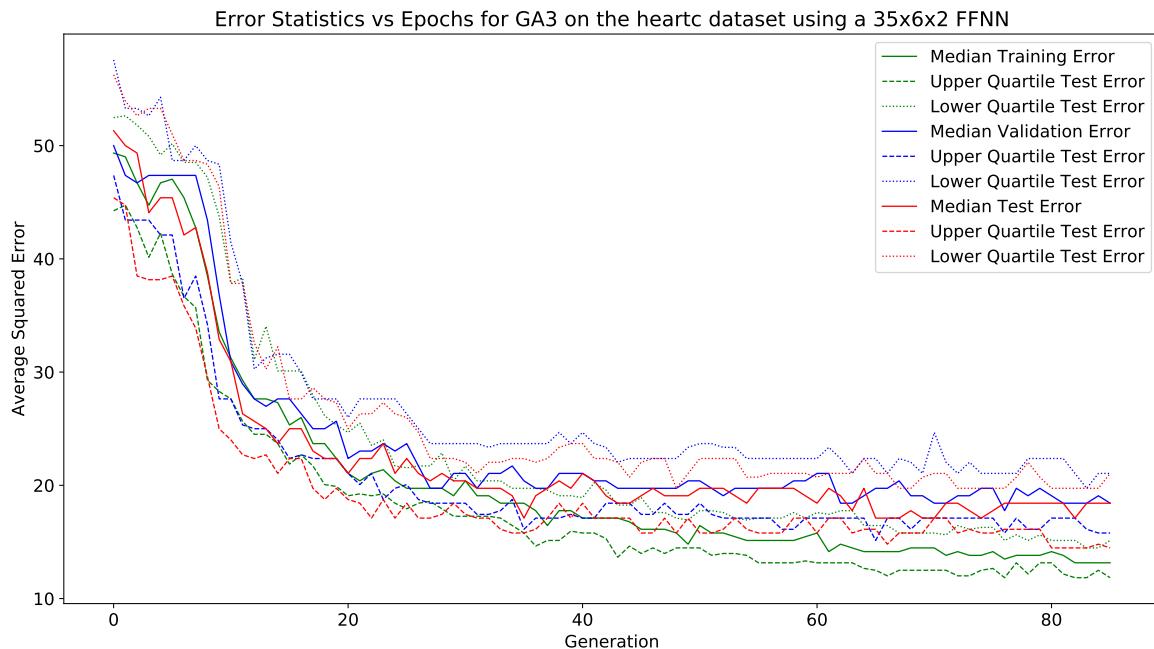
Fig. 470 shows the average and standard deviation of the test, training and validation errors. Fig. 471 shows the median and quartiles of the test, training and validation errors based on the best epoch. Finally, Fig. 472 shows the median and quartiles of the test, training and validation errors per mini-batch.



**Figure 470.** Graph of mean and standard deviation of errors vs epochs



**Figure 471.** Graph of test error vs epochs for the gradient based algorithms



**Figure 472.** Graph of test error vs epochs for the gradient based algorithms

### 16.6.19 iRPROP-

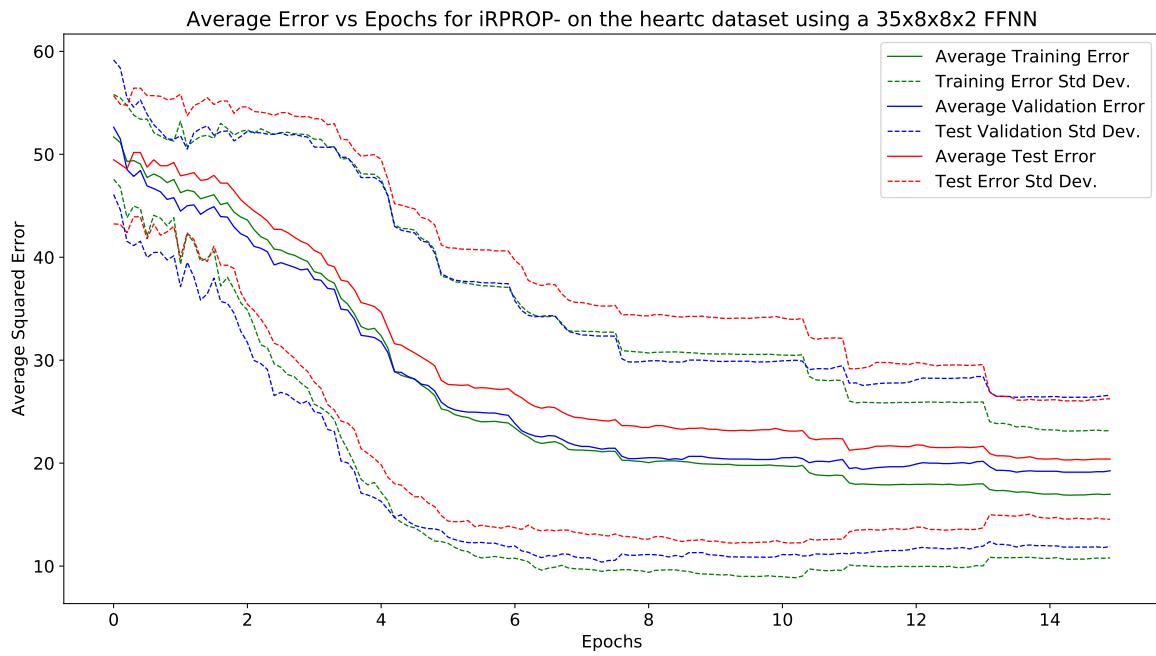
This section displays the results obtained using the iRPROP- algorithm.

### 16.6.20 heartc

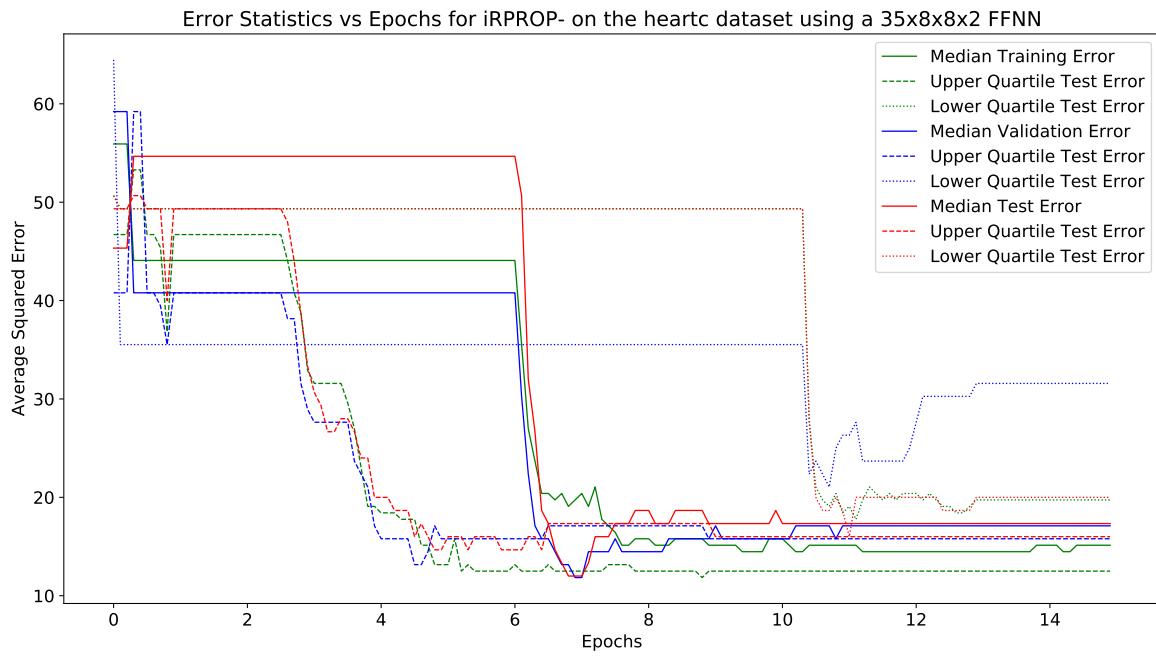
This section displays the results obtained using the heartc algorithm.

#### 35 × 8 × 8 × 2 Architecture:

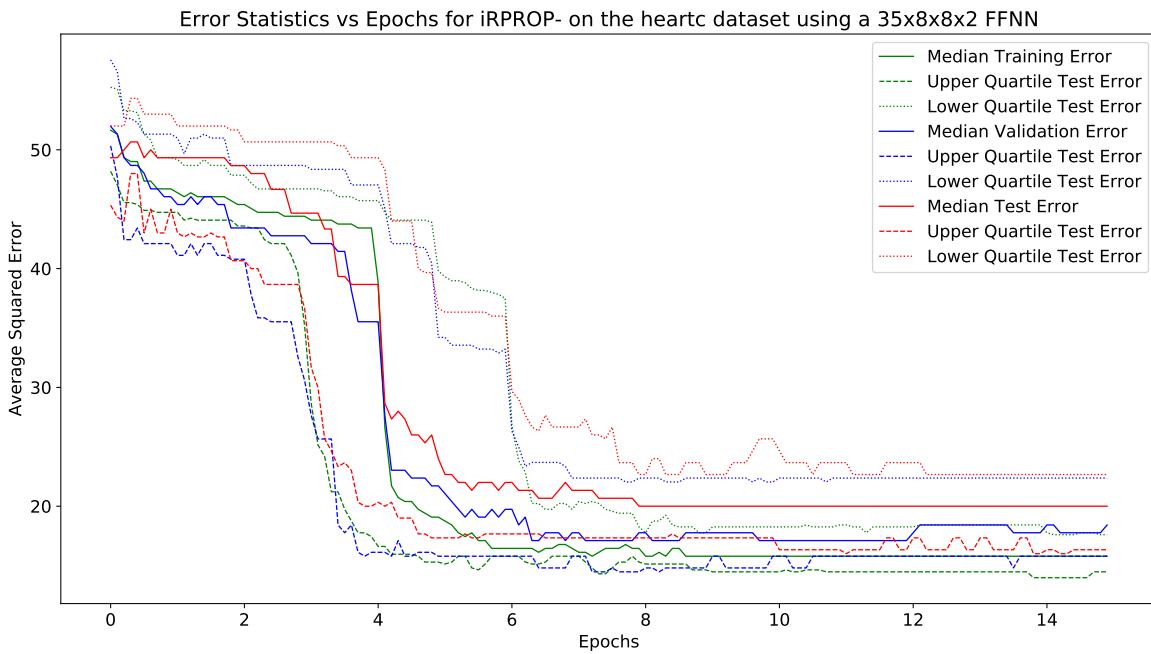
Fig. 473 shows the average and standard deviation of the test, training and validation errors. Fig. 474 shows the median and quartiles of the test, training and validation errors based on the best epoch. Finally, Fig. 475 shows the median and quartiles of the test, training and validation errors per mini-batch.



**Figure 473.** Graph of mean and standard deviation of errors vs epochs



**Figure 474.** Graph of test error vs epochs for the gradient based algorithms



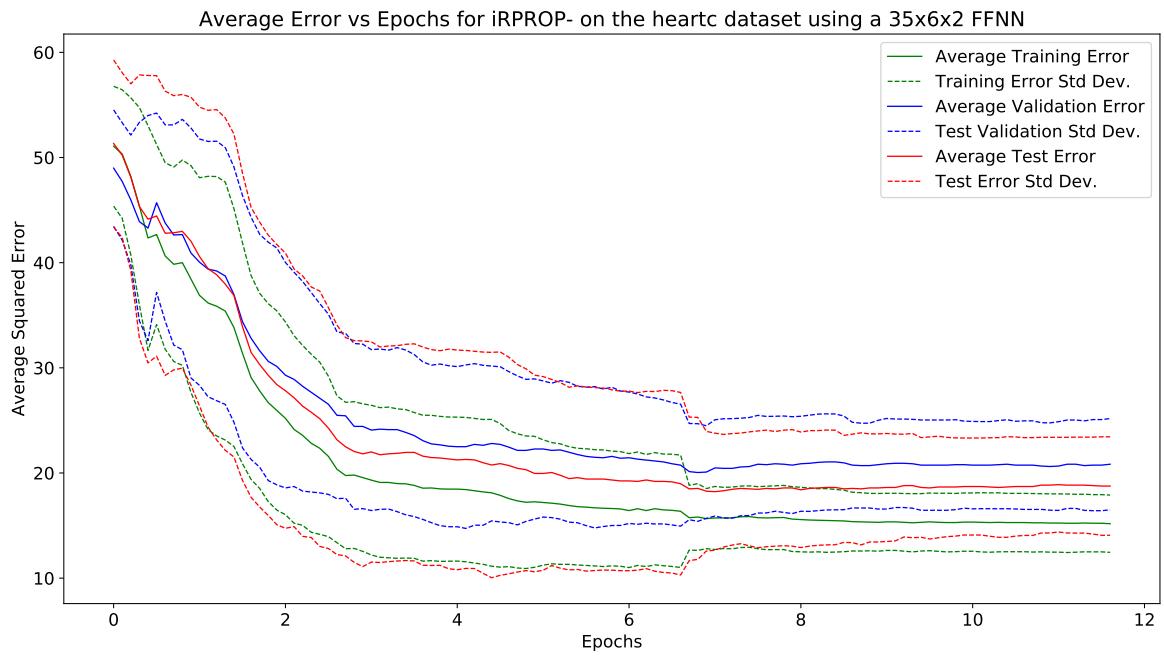
**Figure 475.** Graph of test error vs epochs for the gradient based algorithms

### 16.6.21 heartc

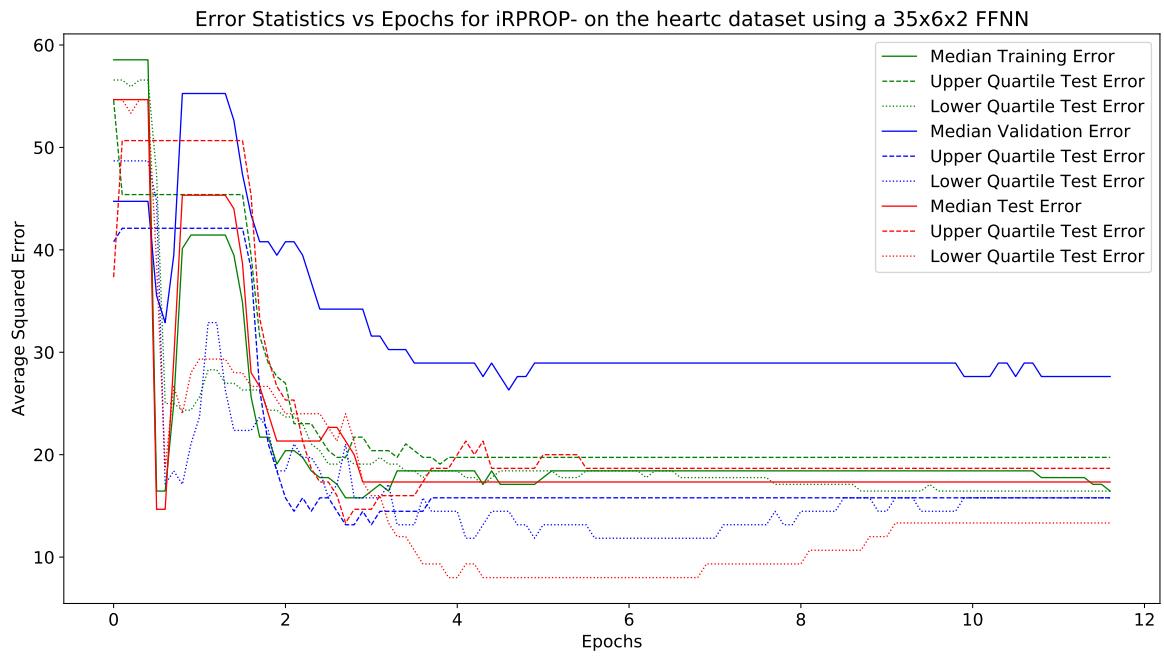
This section displays the results obtained using the heartc algorithm.

#### 35 × 6 × 2 Architecture:

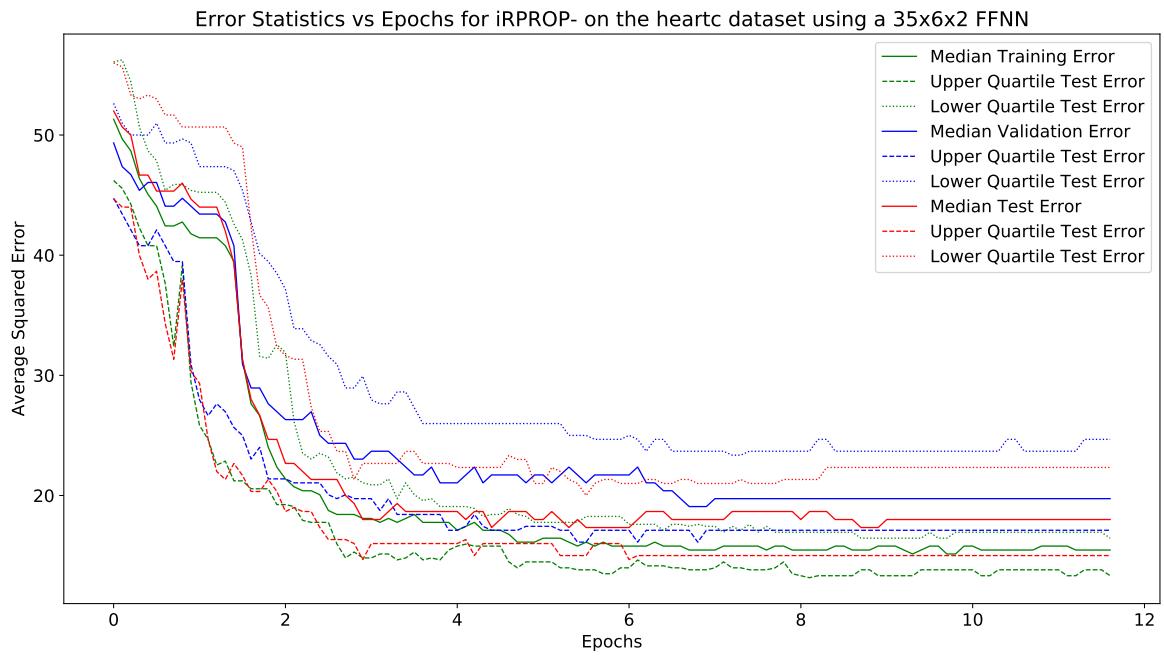
Fig. 476 shows the average and standard deviation of the test, training and validation errors. Fig. 477 shows the median and quartiles of the test, training and validation errors based on the best epoch. Finally, Fig. 478 shows the median and quartiles of the test, training and validation errors per mini-batch.



**Figure 476.** Graph of mean and standard deviation of errors vs epochs



**Figure 477.** Graph of test error vs epochs for the gradient based algorithms



**Figure 478.** Graph of test error vs epochs for the gradient based algorithms

### 16.6.22 *iRPROP+*

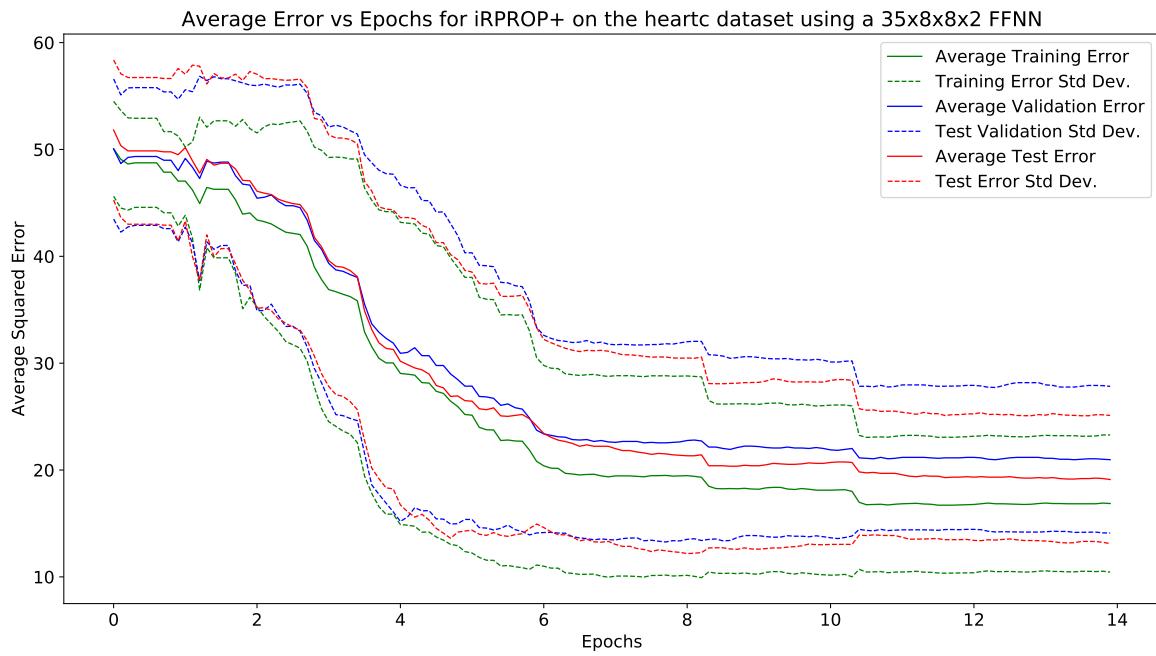
This section displays the results obtained using the iRPROP+ algorithm.

### 16.6.23 *heartc*

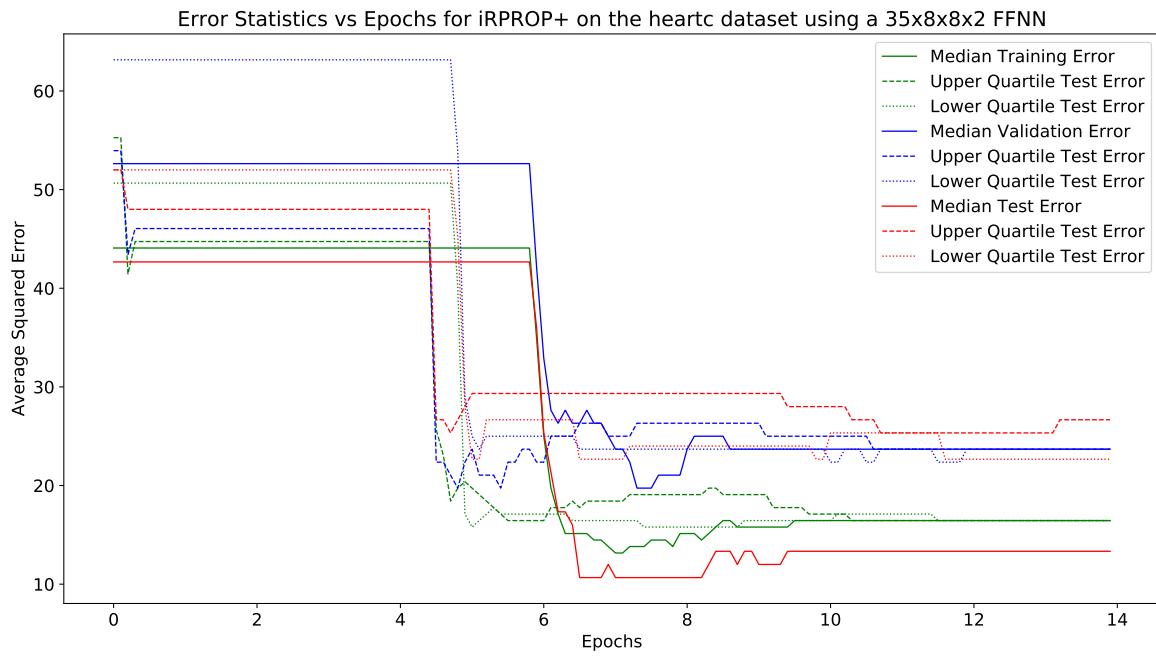
This section displays the results obtained using the heartc algorithm.

#### 35 × 8 × 8 × 2 Architecture:

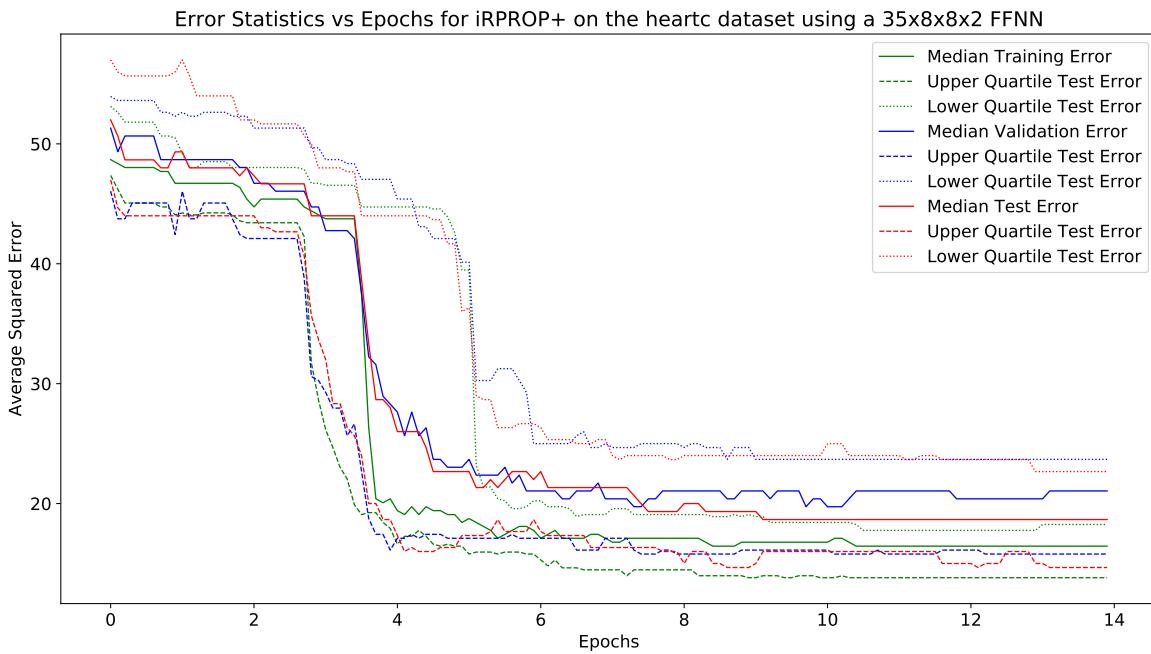
Fig. 479 shows the average and standard deviation of the test, training and validation errors. Fig. 480 shows the median and quartiles of the test, training and validation errors based on the best epoch. Finally, Fig. 481 shows the median and quartiles of the test, training and validation errors per mini-batch.



**Figure 479.** Graph of mean and standard deviation of errors vs epochs



**Figure 480.** Graph of test error vs epochs for the gradient based algorithms



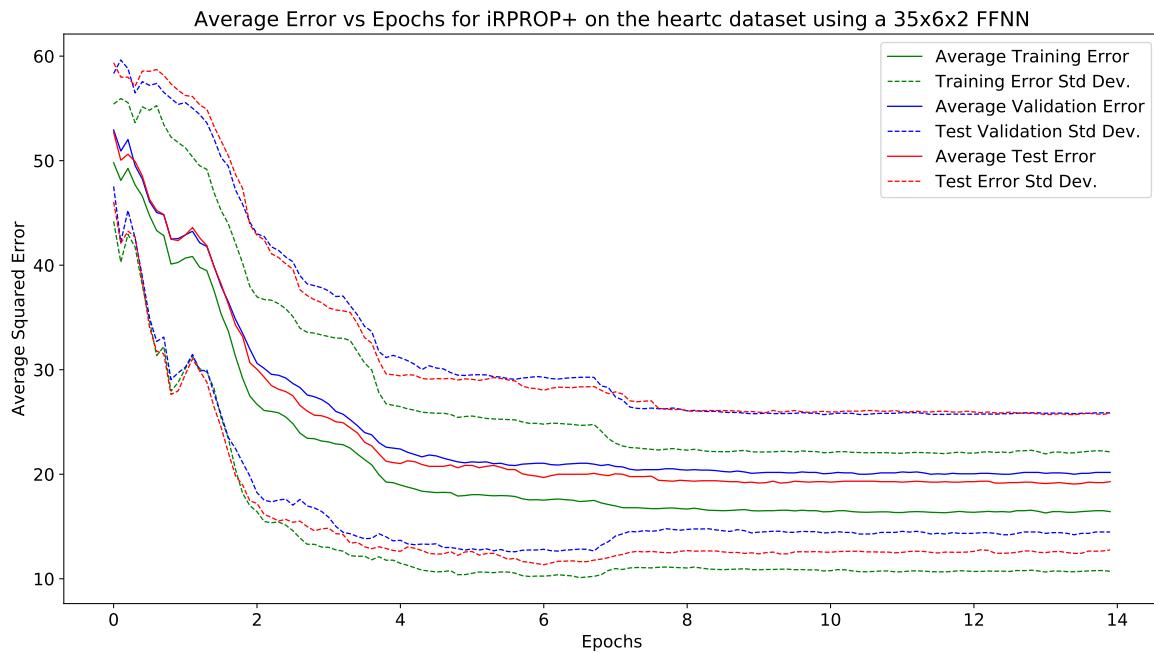
**Figure 481.** Graph of test error vs epochs for the gradient based algorithms

### 16.6.24 heartc

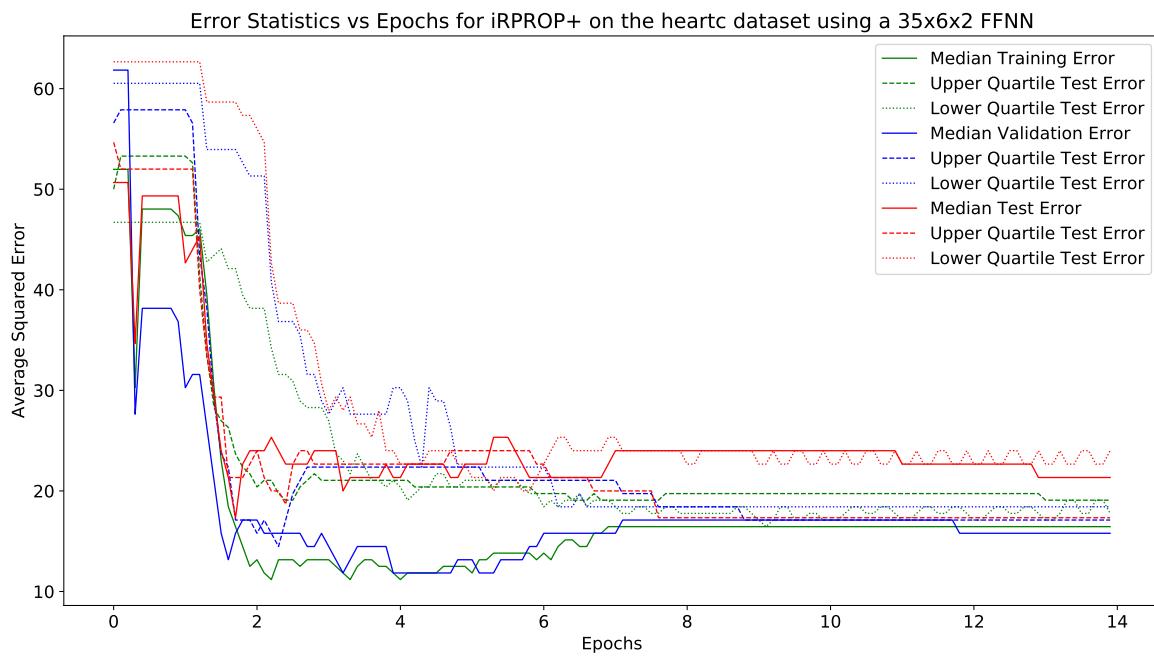
This section displays the results obtained using the heartc algorithm.

#### 35 × 6 × 2 Architecture:

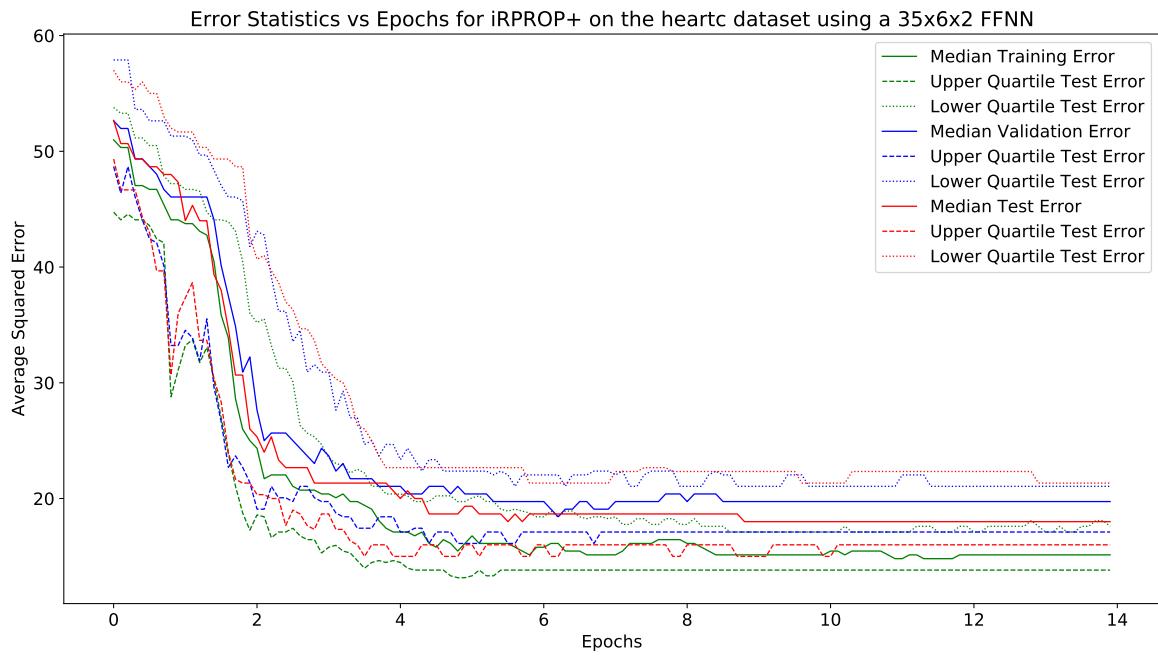
Fig. 482 shows the average and standard deviation of the test, training and validation errors. Fig. 483 shows the median and quartiles of the test, training and validation errors based on the best epoch. Finally, Fig. 484 shows the median and quartiles of the test, training and validation errors per mini-batch.



**Figure 482.** Graph of mean and standard deviation of errors vs epochs



**Figure 483.** Graph of test error vs epochs for the gradient based algorithms



**Figure 484.** Graph of test error vs epochs for the gradient based algorithms

### 16.6.25 PSO

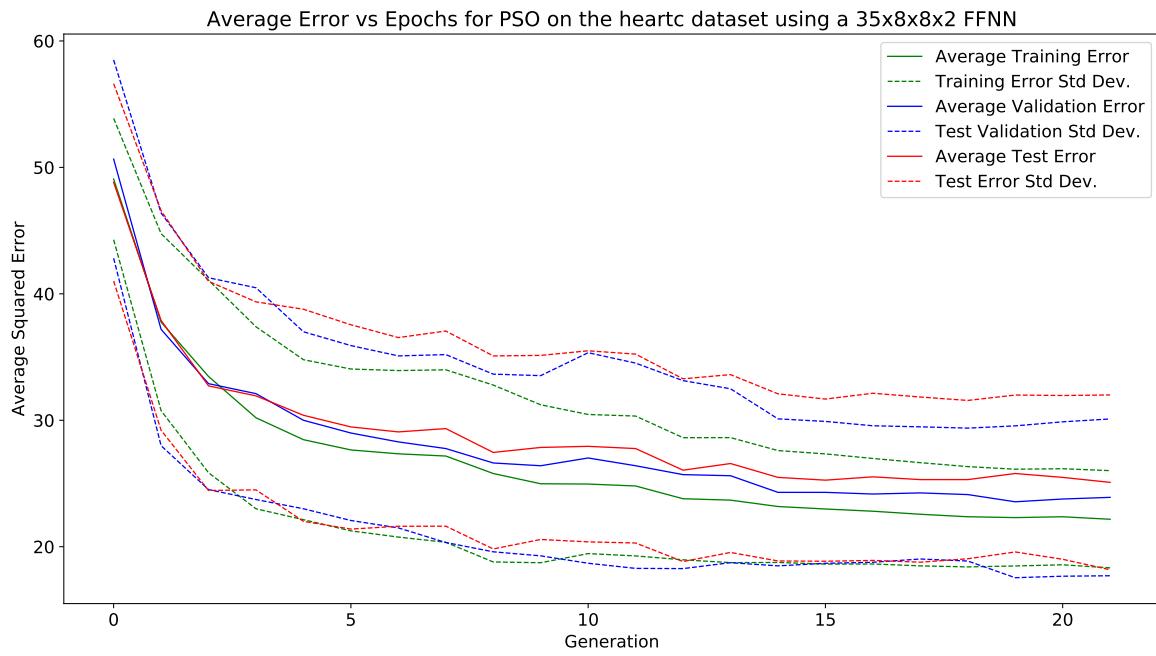
This section displays the results obtained using the PSO algorithm.

### 16.6.26 heartc

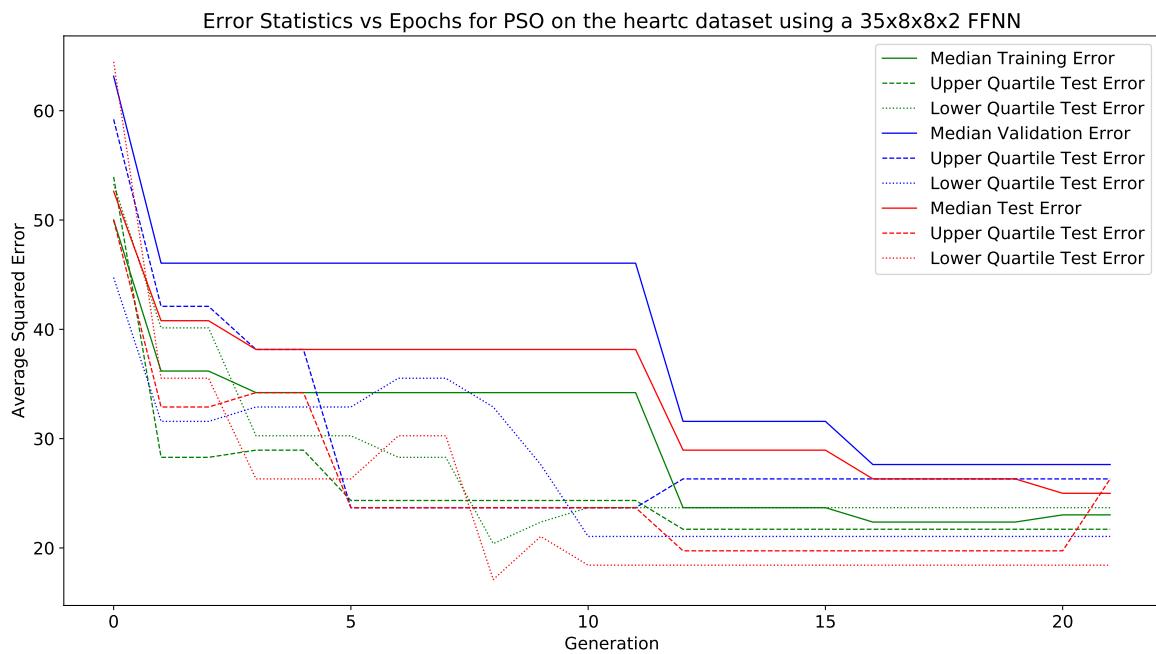
This section displays the results obtained using the heartc algorithm.

#### 35 × 8 × 8 × 2 Architecture:

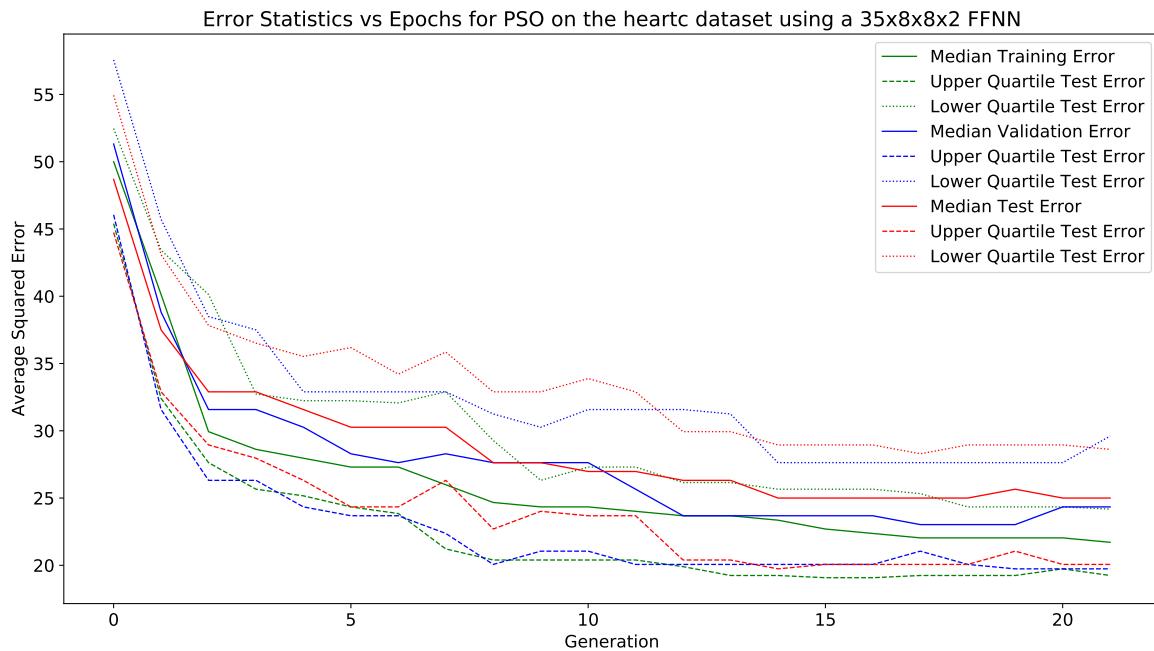
Fig. 485 shows the average and standard deviation of the test, training and validation errors. Fig. 486 shows the median and quartiles of the test, training and validation errors based on the best epoch. Finally, Fig. 487 shows the median and quartiles of the test, training and validation errors per mini-batch.



**Figure 485.** Graph of mean and standard deviation of errors vs epochs



**Figure 486.** Graph of test error vs epochs for the gradient based algorithms



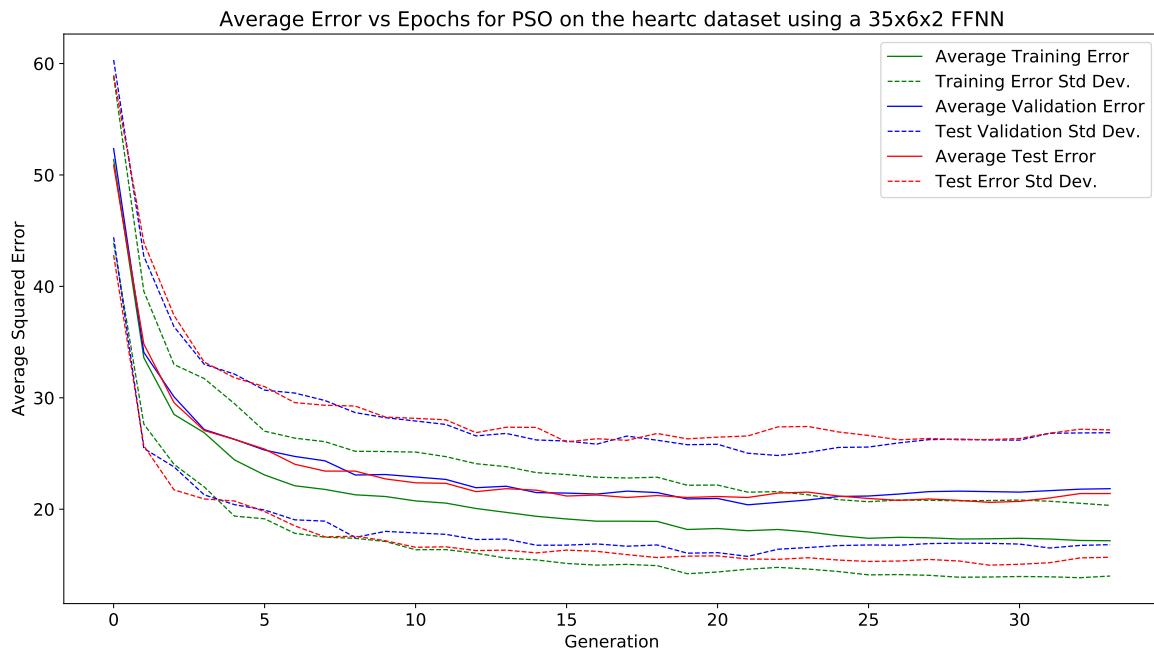
**Figure 487.** Graph of test error vs epochs for the gradient based algorithms

### 16.6.27 heartc

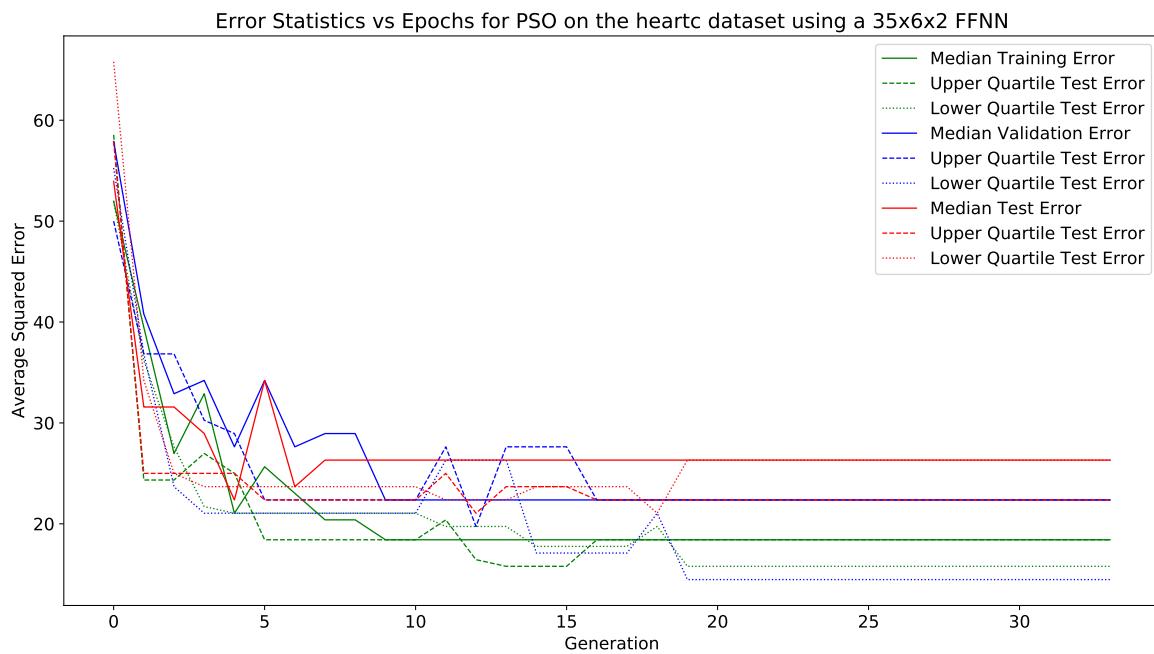
This section displays the results obtained using the heartc algorithm.

#### 35 × 6 × 2 Architecture:

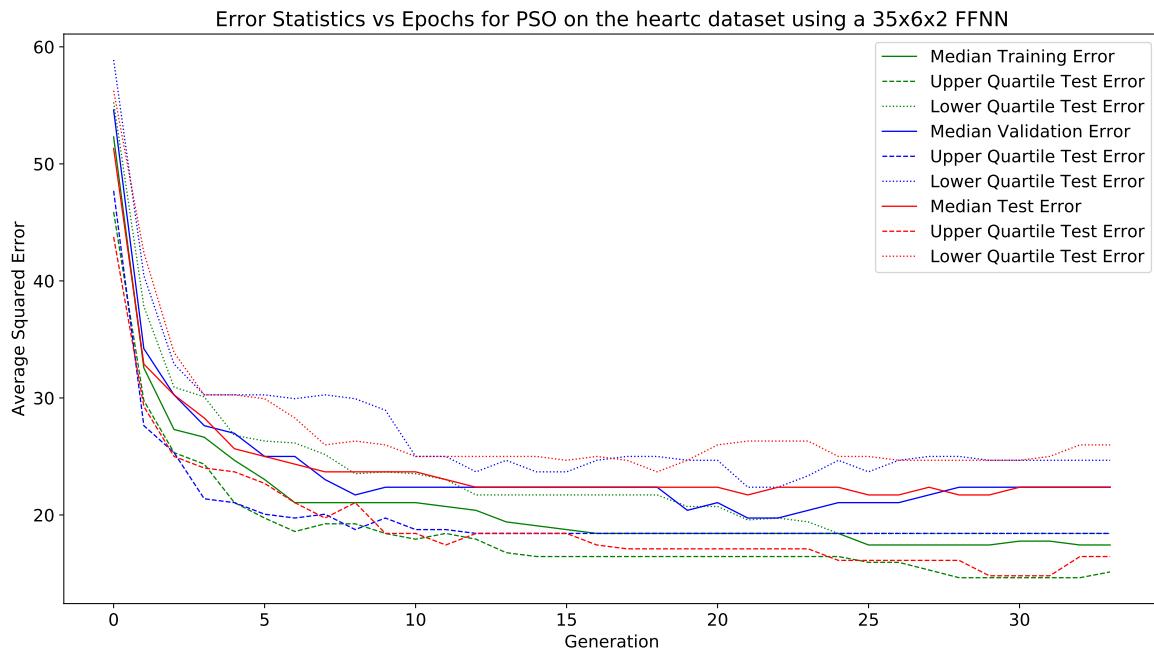
Fig. 488 shows the average and standard deviation of the test, training and validation errors. Fig. 489 shows the median and quartiles of the test, training and validation errors based on the best epoch. Finally, Fig. 490 shows the median and quartiles of the test, training and validation errors per mini-batch.



**Figure 488.** Graph of mean and standard deviation of errors vs epochs



**Figure 489.** Graph of test error vs epochs for the gradient based algorithms



**Figure 490.** Graph of test error vs epochs for the gradient based algorithms

### 16.6.28 QuickProp

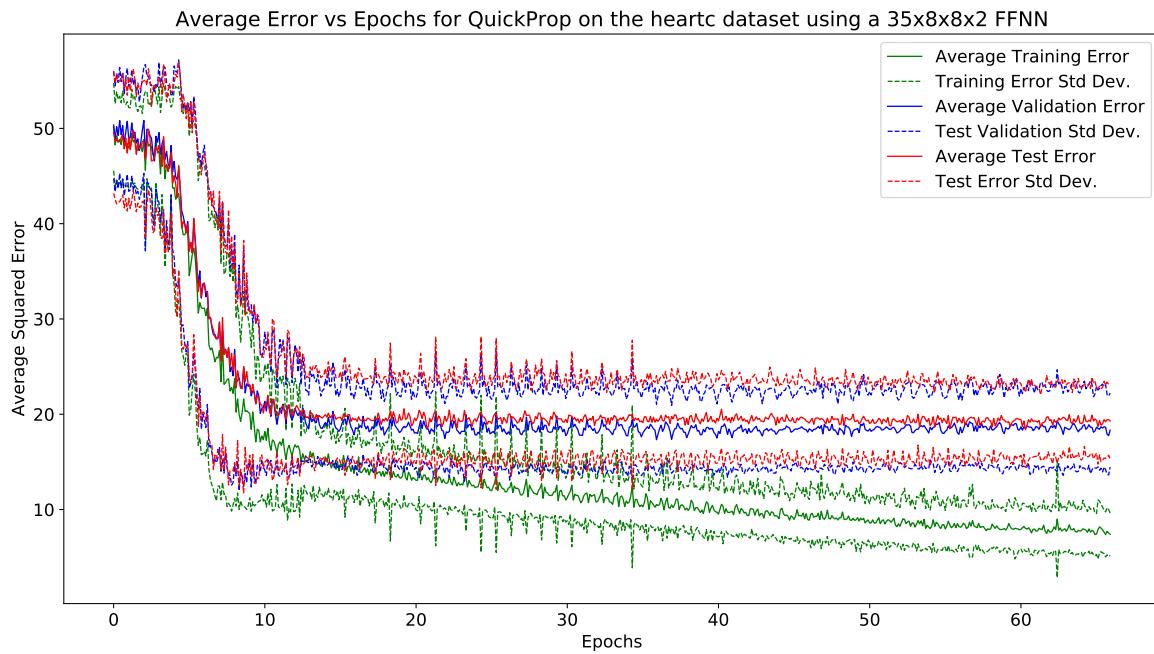
This section displays the results obtained using the QuickProp algorithm.

### 16.6.29 heartc

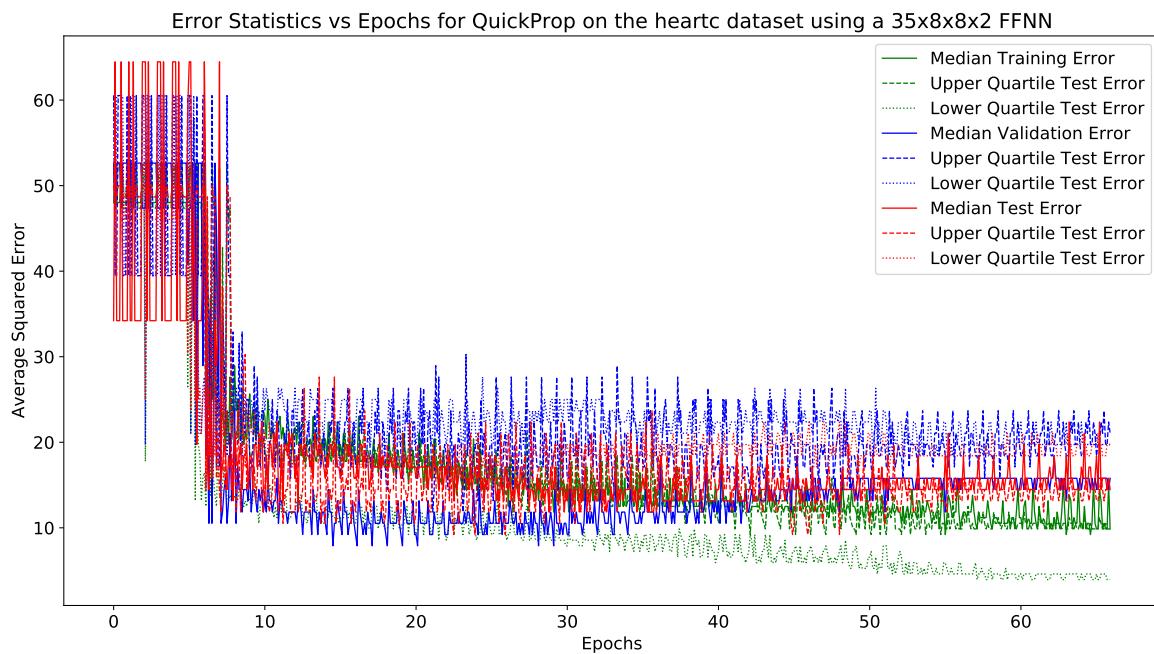
This section displays the results obtained using the heartc algorithm.

#### 35 × 8 × 8 × 2 Architecture:

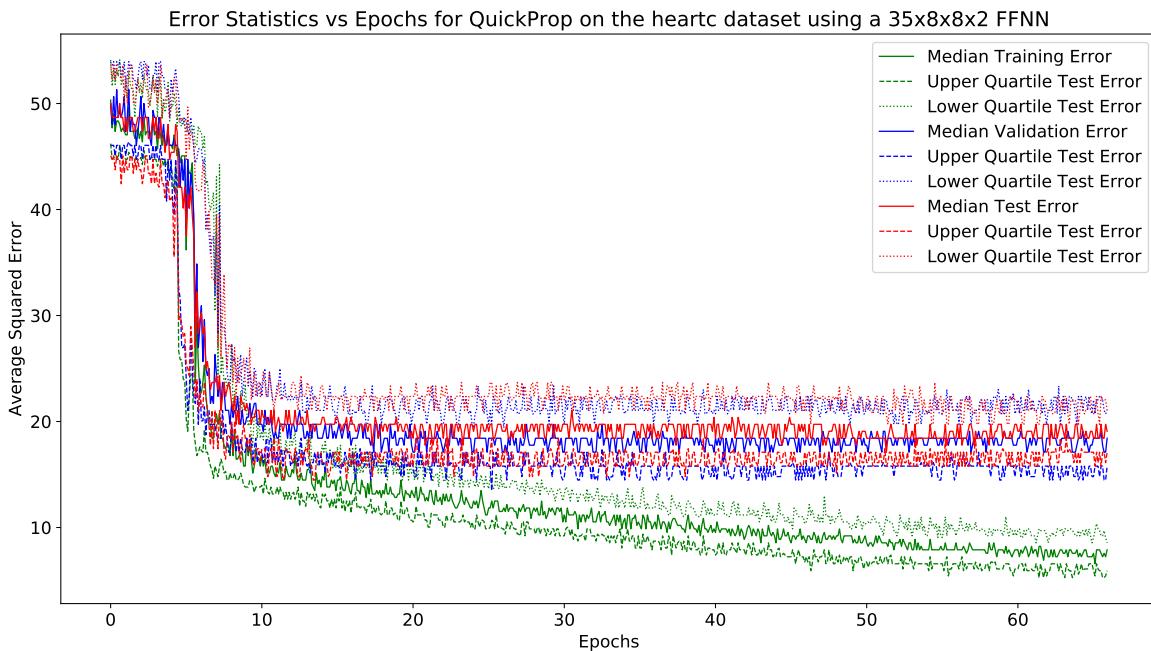
Fig. 491 shows the average and standard deviation of the test, training and validation errors. Fig. 492 shows the median and quartiles of the test, training and validation errors based on the best epoch. Finally, Fig. 493 shows the median and quartiles of the test, training and validation errors per mini-batch.



**Figure 491.** Graph of mean and standard deviation of errors vs epochs



**Figure 492.** Graph of test error vs epochs for the gradient based algorithms



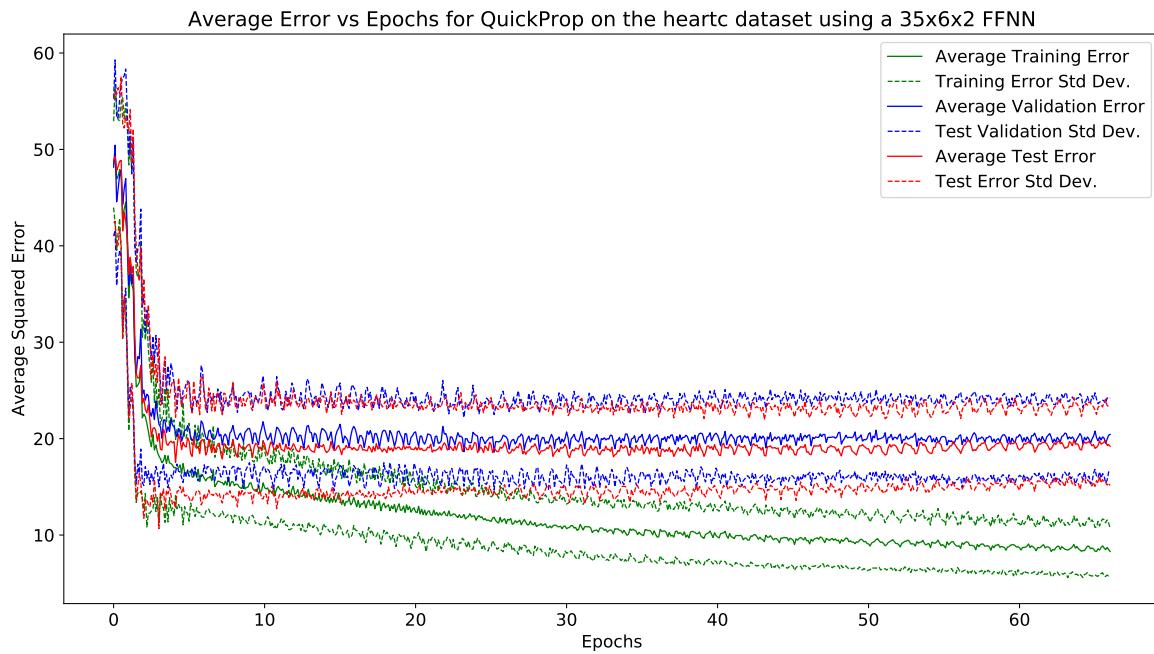
**Figure 493.** Graph of test error vs epochs for the gradient based algorithms

### 16.6.30 heartc

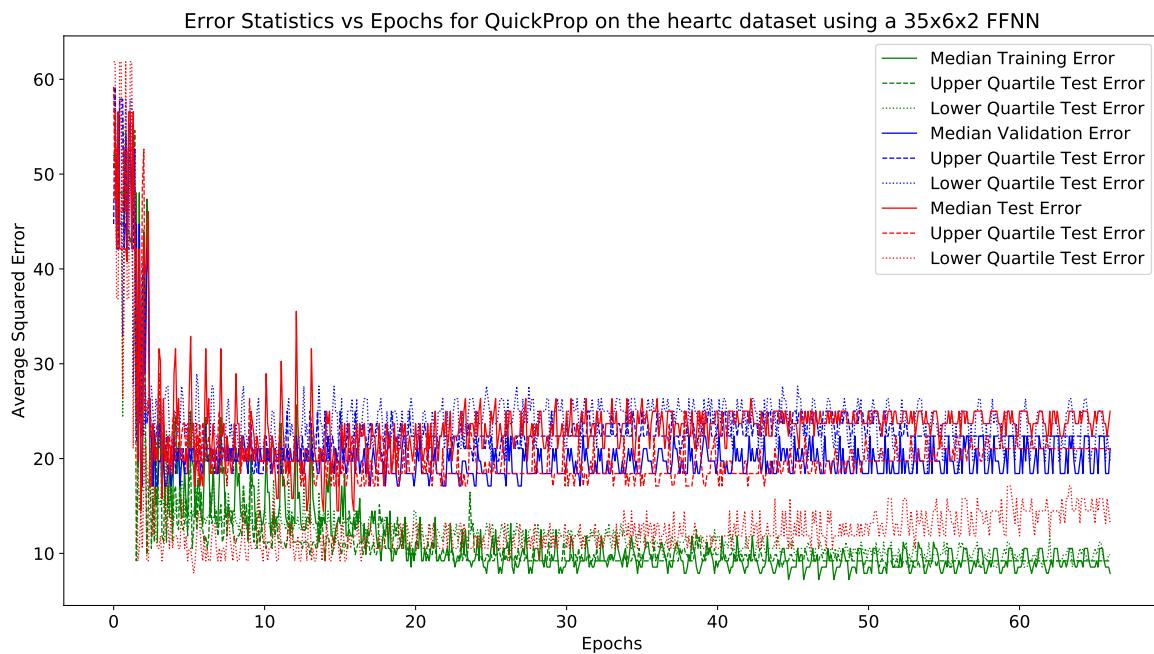
This section displays the results obtained using the heartc algorithm.

#### 35 × 6 × 2 Architecture:

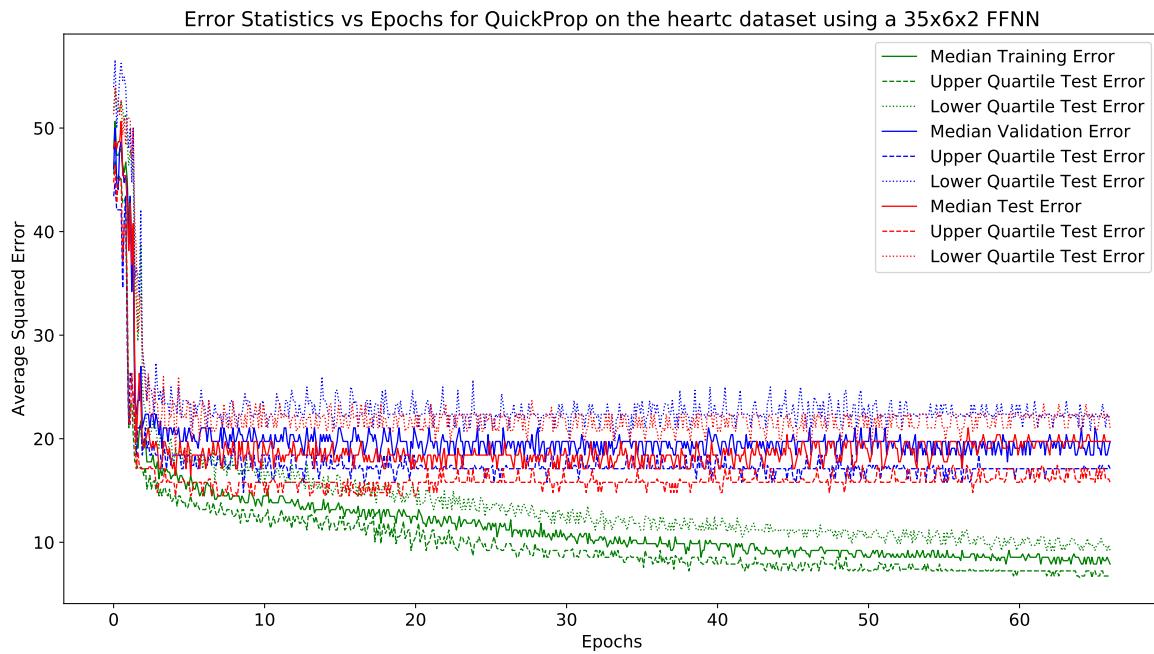
Fig. 494 shows the average and standard deviation of the test, training and validation errors. Fig. 495 shows the median and quartiles of the test, training and validation errors based on the best epoch. Finally, Fig. 496 shows the median and quartiles of the test, training and validation errors per mini-batch.



**Figure 494.** Graph of mean and standard deviation of errors vs epochs



**Figure 495.** Graph of test error vs epochs for the gradient based algorithms



**Figure 496.** Graph of test error vs epochs for the gradient based algorithms

### 16.6.31 RPROP-

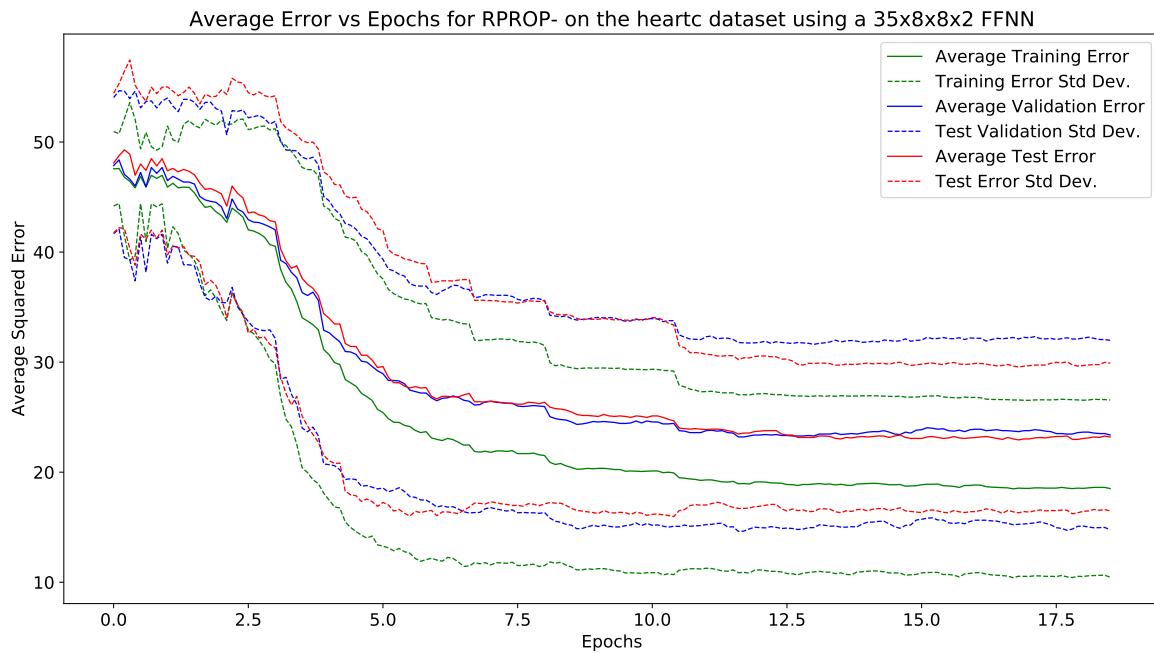
This section displays the results obtained using the RPROP- algorithm.

### 16.6.32 heartc

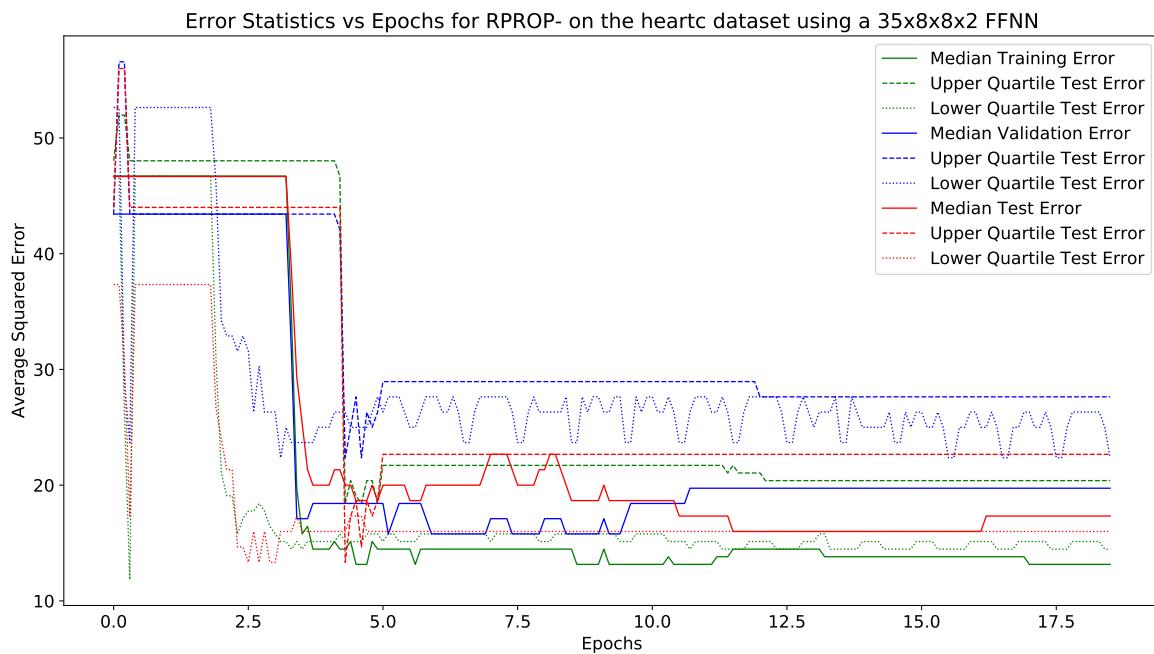
This section displays the results obtained using the heartc algorithm.

#### 35 × 8 × 8 × 2 Architecture:

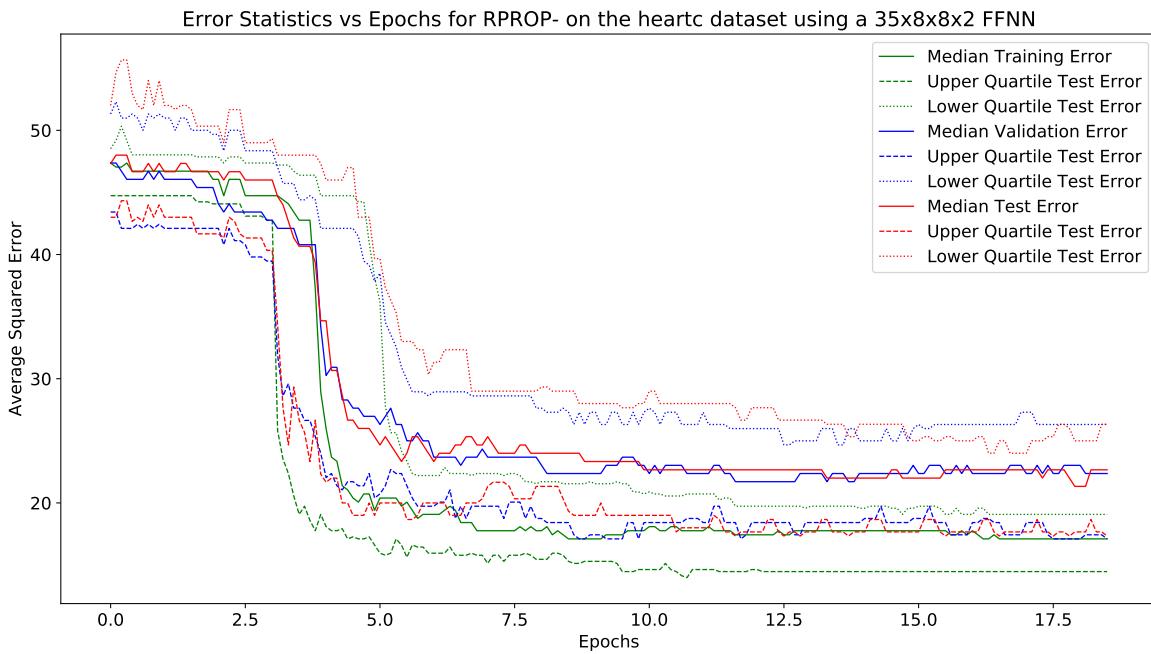
Fig. 497 shows the average and standard deviation of the test, training and validation errors. Fig. 498 shows the median and quartiles of the test, training and validation errors based on the best epoch. Finally, Fig. 499 shows the median and quartiles of the test, training and validation errors per mini-batch.



**Figure 497.** Graph of mean and standard deviation of errors vs epochs



**Figure 498.** Graph of test error vs epochs for the gradient based algorithms



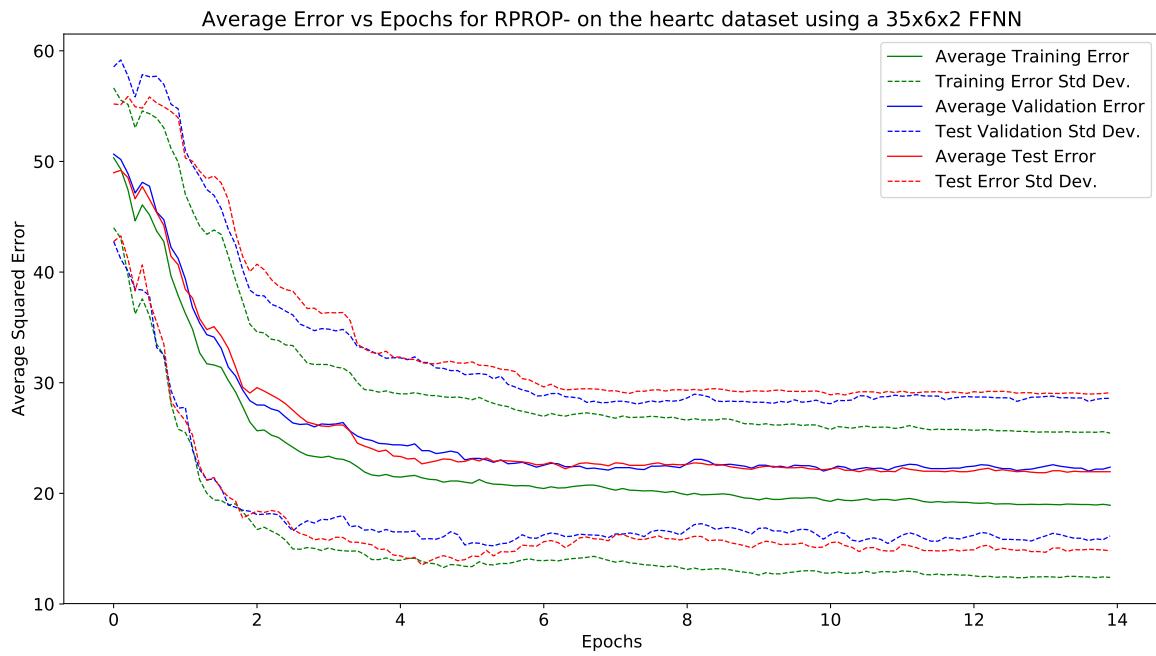
**Figure 499.** Graph of test error vs epochs for the gradient based algorithms

### 16.6.33 heartc

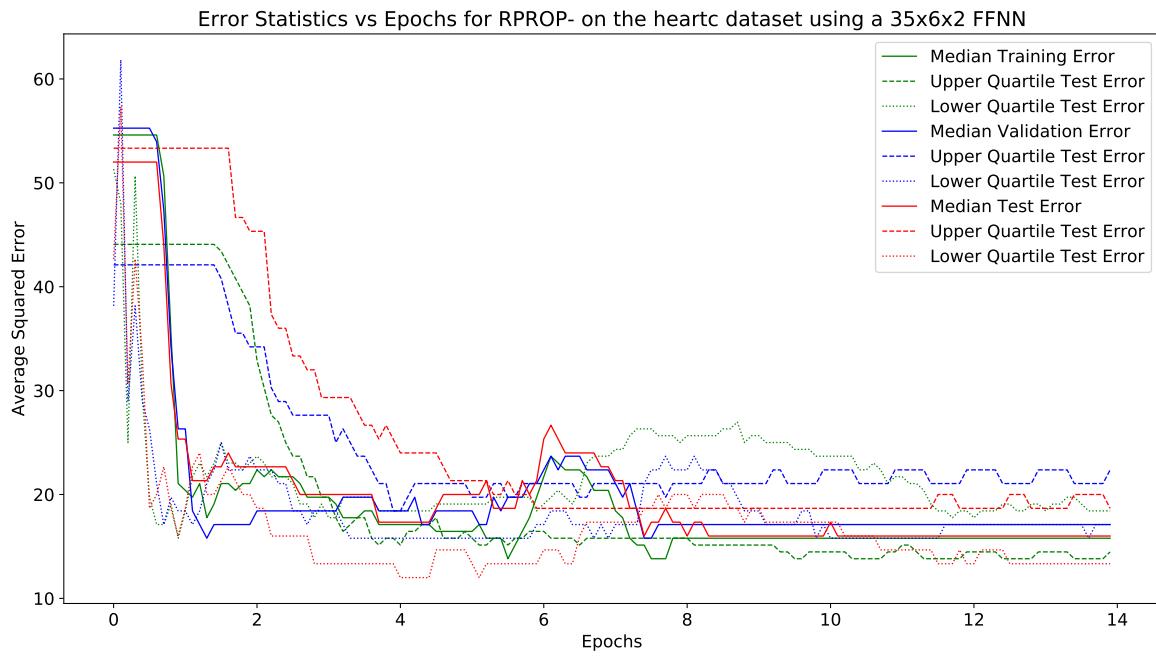
This section displays the results obtained using the heartc algorithm.

#### 35 × 6 × 2 Architecture:

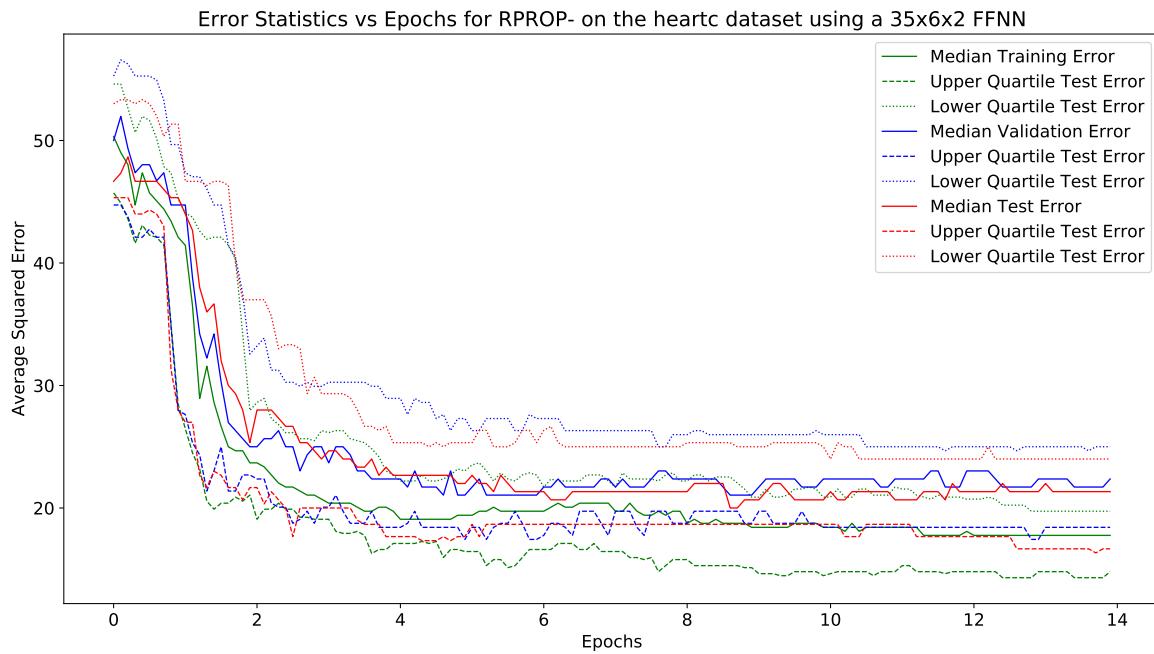
Fig. 500 shows the average and standard deviation of the test, training and validation errors. Fig. 501 shows the median and quartiles of the test, training and validation errors based on the best epoch. Finally, Fig. 502 shows the median and quartiles of the test, training and validation errors per mini-batch.



**Figure 500.** Graph of mean and standard deviation of errors vs epochs



**Figure 501.** Graph of test error vs epochs for the gradient based algorithms



**Figure 502.** Graph of test error vs epochs for the gradient based algorithms

### 16.6.34 RPROP+

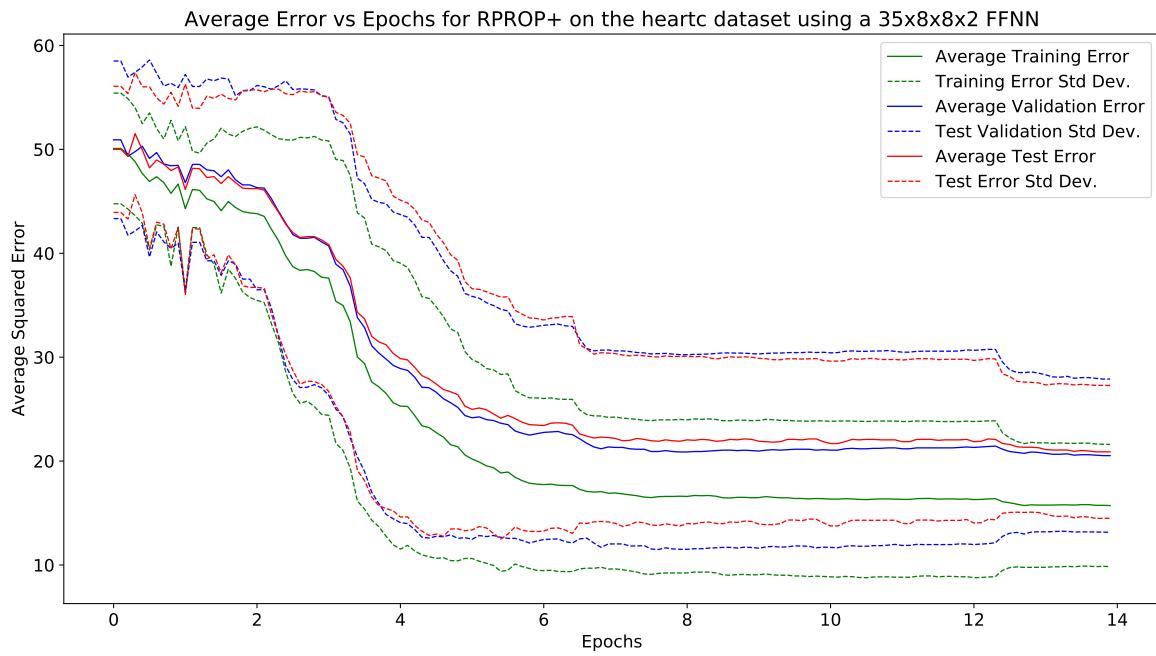
This section displays the results obtained using the RPROP+ algorithm.

### 16.6.35 heartc

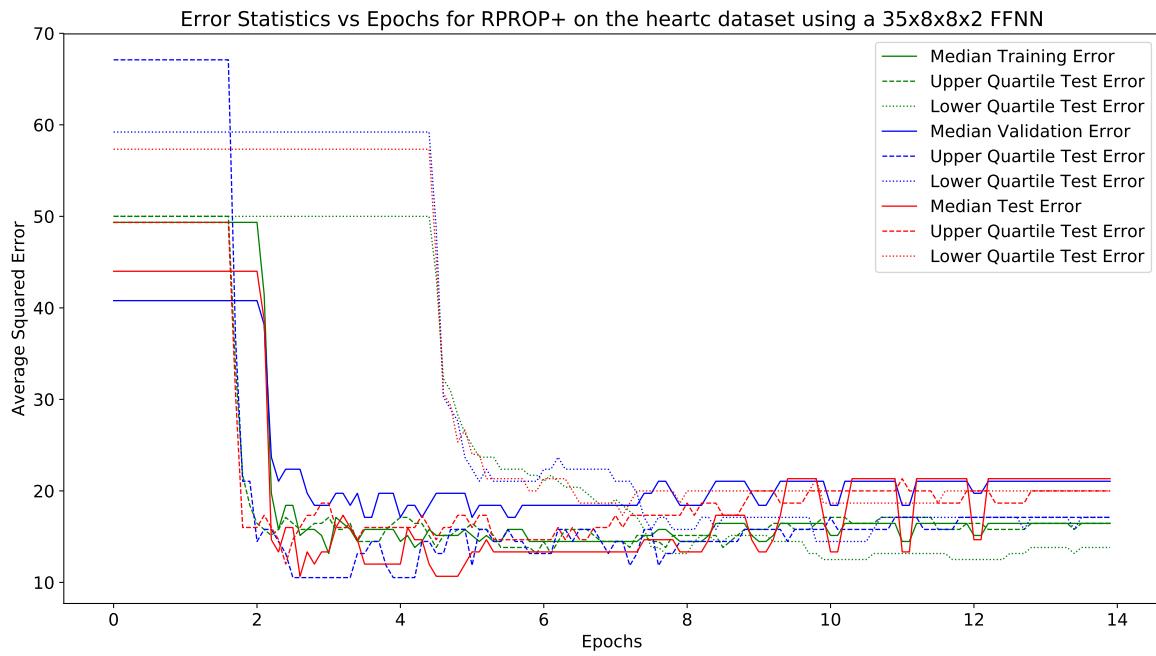
This section displays the results obtained using the heartc algorithm.

#### 35 × 8 × 8 × 2 Architecture:

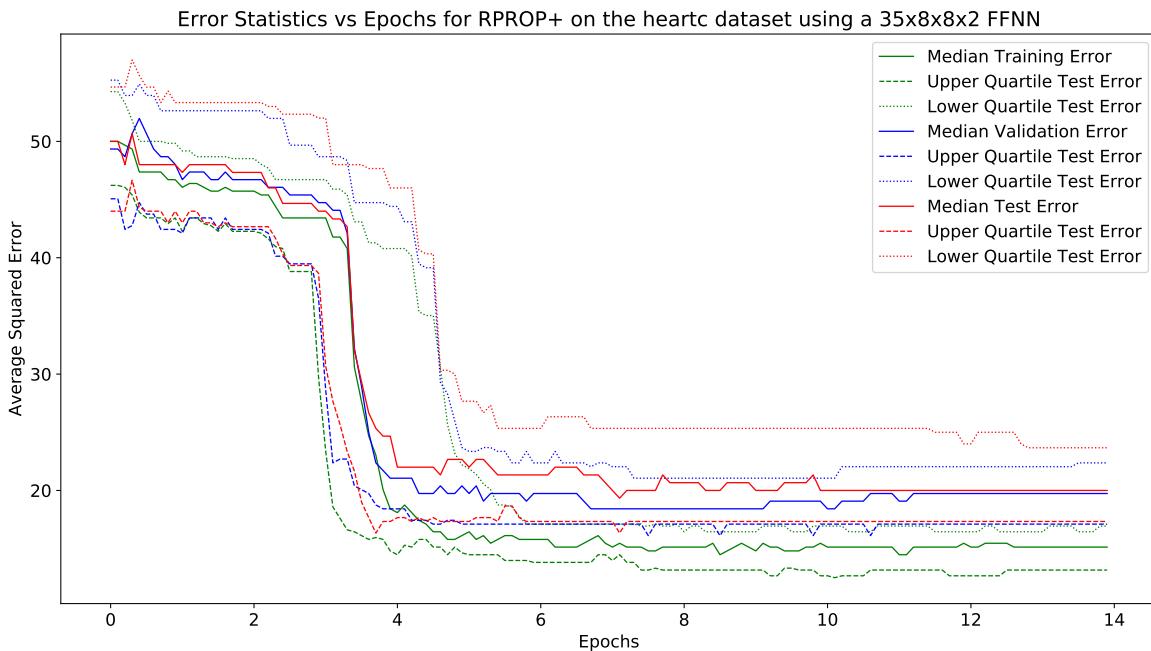
Fig. 503 shows the average and standard deviation of the test, training and validation errors. Fig. 504 shows the median and quartiles of the test, training and validation errors based on the best epoch. Finally, Fig. 505 shows the median and quartiles of the test, training and validation errors per mini-batch.



**Figure 503.** Graph of mean and standard deviation of errors vs epochs



**Figure 504.** Graph of test error vs epochs for the gradient based algorithms



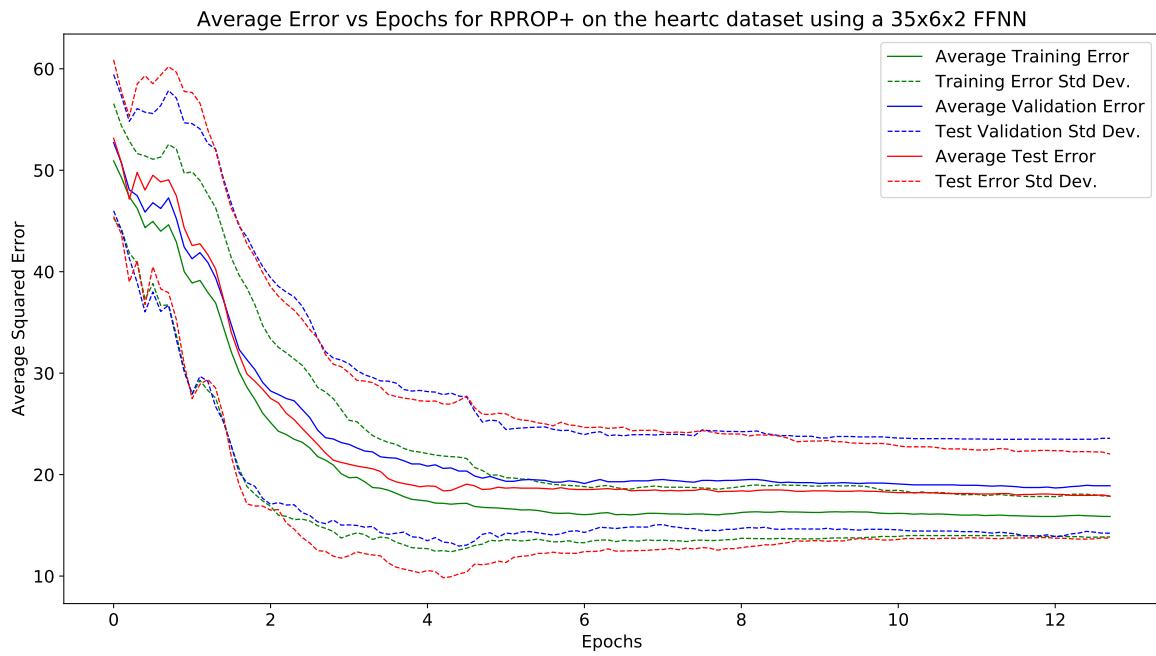
**Figure 505.** Graph of test error vs epochs for the gradient based algorithms

### 16.6.36 heartc

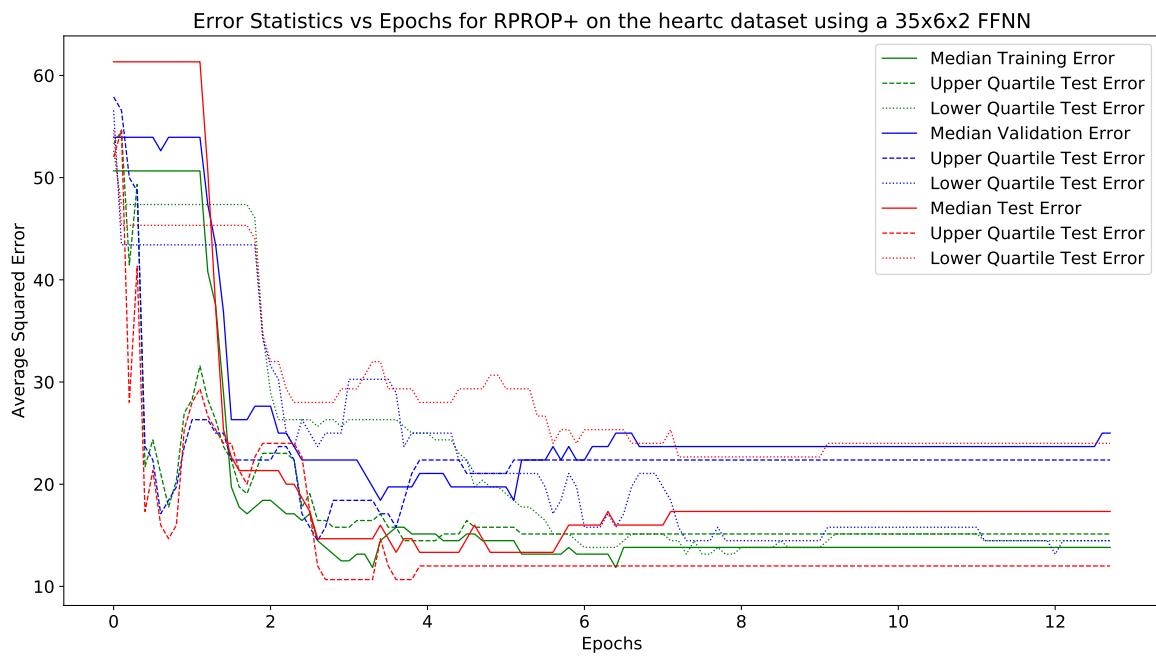
This section displays the results obtained using the heartc algorithm.

#### 35 × 6 × 2 Architecture:

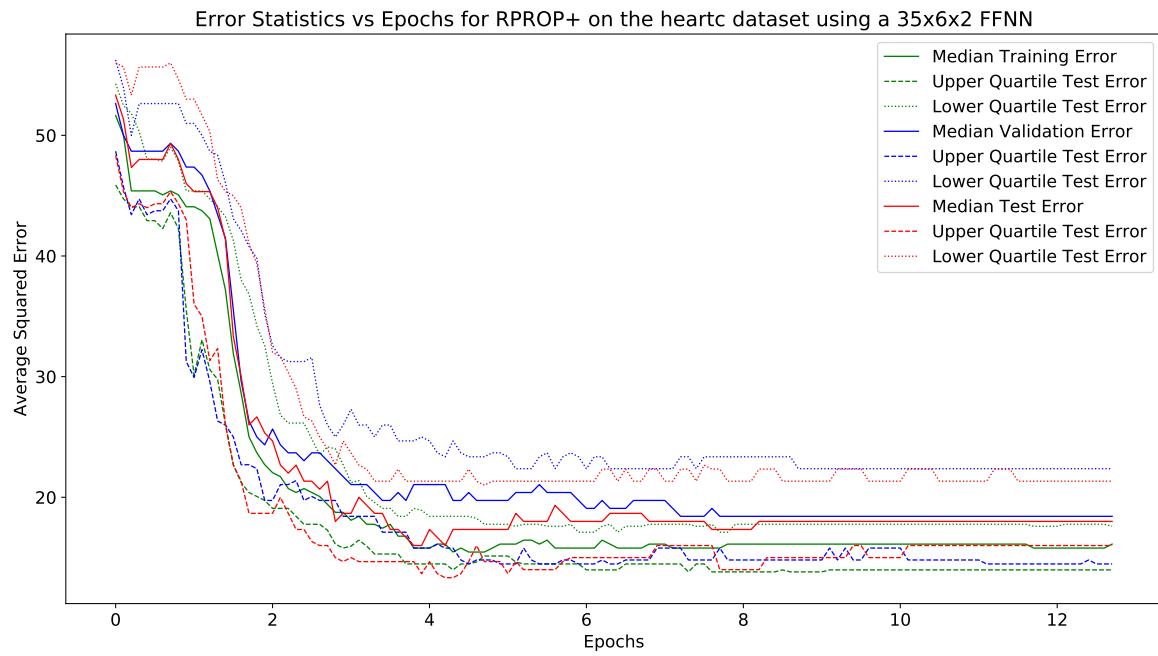
Fig. 506 shows the average and standard deviation of the test, training and validation errors. Fig. 507 shows the median and quartiles of the test, training and validation errors based on the best epoch. Finally, Fig. 508 shows the median and quartiles of the test, training and validation errors per mini-batch.



**Figure 506.** Graph of mean and standard deviation of errors vs epochs



**Figure 507.** Graph of test error vs epochs for the gradient based algorithms



**Figure 508.** Graph of test error vs epochs for the gradient based algorithms