



# TMVA Optimizers

- Ravi Kiran S

# Work done:

- Implemented various optimizers including SGD, Adam, Adagrad, Adadelata, RMSProp.
- Implemented various operations needed for performing the updates like ConstAdd, ConstMult, SquareElementWise, SqrtElementWise, ReciprocalElementWise in Reference, CPU and GPU architectures.
- Implemented a new evaluation metric called meanAbsoluteError() between two matrices which is used for testing the optimizers.
- Finally, after getting reviewed from Stefan and Lorenzo, Got all the PRs merged into root/master successfully :D

# TMVA Optimizers - Overview:

```

TRMSProp
#Momentum: Scalar_t
#rho: Scalar_t
#Epsilon: Scalar_t
#PastSquaredWeightGradients: vector<vector<Matrix_D>>
#PastSquaredBiasGradients: vector<vector<Matrix_D>>
#WeightUpdates: vector<vector<Matrix_D>>
#BiasUpdates: vector<vector<Matrix_D>>

+TRMSProp(deepNet: DeepNet_t &, learningRate: Scalar_t = 0.001, momentum: Scalar_t = 0.0, rho: Scalar_t = 0.9, epsilon: Scalar_t = 0.000001)
#UpdateWeights(layerIndex: size_t, weights: vector<Matrix_D> &, weightGradients: const vector<Matrix_D> &); void
#UpdateBiases(layerIndex: size_t, biases: vector<Matrix_D> &, biasGradients: const vector<Matrix_D> &); void
    
```

```

TSGD
#Momentum: Scalar_t
#PastWeightGradients: vector<vector<Matrix_D>>
#PastBiasGradients: vector<vector<Matrix_D>>

+TSGD(learningRate: Scalar_t = 0.001, deepNet: DeepNet_t &, momentum: Scalar_t = 0.0)
#UpdateWeights(layerIndex: size_t, weights: vector<Matrix_D> &, weightGradients: const vector<Matrix_D> &); void
#UpdateBiases(layerIndex: size_t, biases: vector<Matrix_D> &, biasGradients: const vector<Matrix_D> &); void
    
```

```

VOptimizer
#learningRate: Scalar_t
#GlobalStep: size_t
#DeepNet: DeepNet_t&

+VOptimize(learningRate: Scalar_t, deepNet: DeepNet_t&)
+Step(); void
+IncrementGlobalStep(); void
#UpdateWeights(layerIndex: size_t, weights: vector<Matrix_D> &, weightGradients: const vector<Matrix_D> &); virtual void
#UpdateBiases(layerIndex: size_t, biases: vector<Matrix_D> &, biasGradients: const vector<Matrix_D> &); virtual void
    
```

```

TAdam
#Beta1: Scalar_t
#Beta2: Scalar_t
#Epsilon: Scalar_t
#FirstMomentWeights: vector<vector<Matrix_D>>
#FirstMomentBiases: vector<vector<Matrix_D>>
#SecondMomentWeights: vector<vector<Matrix_D>>
#SecondMomentBiases: vector<vector<Matrix_D>>

+TAdam(deepNet: DeepNet_t &, learningRate: Scalar_t = 0.001, beta1: Scalar_t = 0.9, beta2: Scalar_t = 0.999, epsilon: Scalar_t = 0.000001)
#UpdateWeights(layerIndex: size_t, weights: vector<Matrix_D> &, weightGradients: const vector<Matrix_D> &); void
#UpdateBiases(layerIndex: size_t, biases: vector<Matrix_D> &, biasGradients: const vector<Matrix_D> &); void
    
```

```

TAdadelta
#rho: Scalar_t
#Epsilon: Scalar_t
#PastSquaredWeightGradients: vector<vector<Matrix_D>>
#PastSquaredBiasGradients: vector<vector<Matrix_D>>
#PastSquaredWeightUpdates: vector<vector<Matrix_D>>
#PastSquaredBiasUpdates: vector<vector<Matrix_D>>

+TAdadelta(deepNet: DeepNet_t &, learningRate: Scalar_t = 1.0, rho: Scalar_t = 0.95, epsilon: Scalar_t = 0.000001)
#UpdateWeights(layerIndex: size_t, weights: vector<Matrix_D> &, weightGradients: const vector<Matrix_D> &); void
#UpdateBiases(layerIndex: size_t, biases: vector<Matrix_D> &, biasGradients: const vector<Matrix_D> &); void
    
```

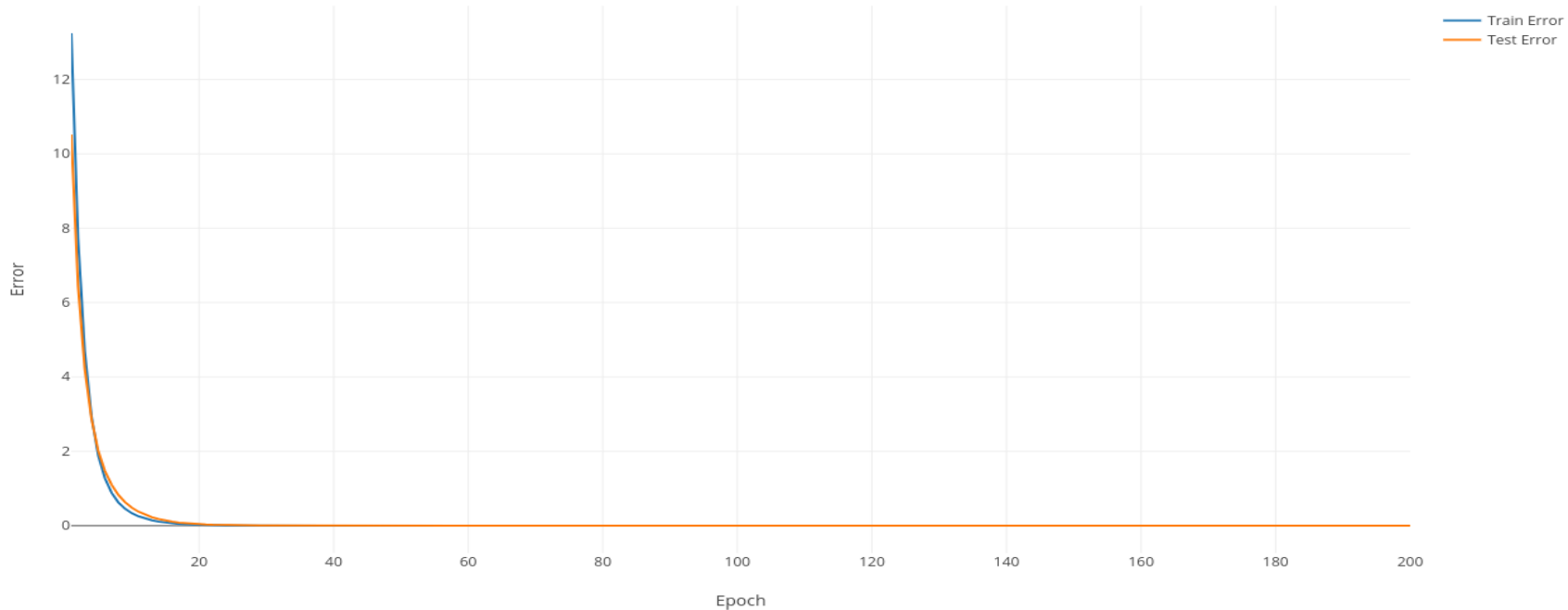
```

TAdagrad
#Epsilon: Scalar_t
#PastSquaredWeightGradients: vector<vector<Matrix_D>>
#PastSquaredBiasGradients: vector<vector<Matrix_D>>

+TAdagrad(deepNet: DeepNet_t &, learningRate: Scalar_t = 0.01, epsilon: Scalar_t = 0.000001)
#UpdateWeights(layerIndex: size_t, weights: vector<Matrix_D> &, weightGradients: const vector<Matrix_D> &); void
#UpdateBiases(layerIndex: size_t, biases: vector<Matrix_D> &, biasGradients: const vector<Matrix_D> &); void
    
```

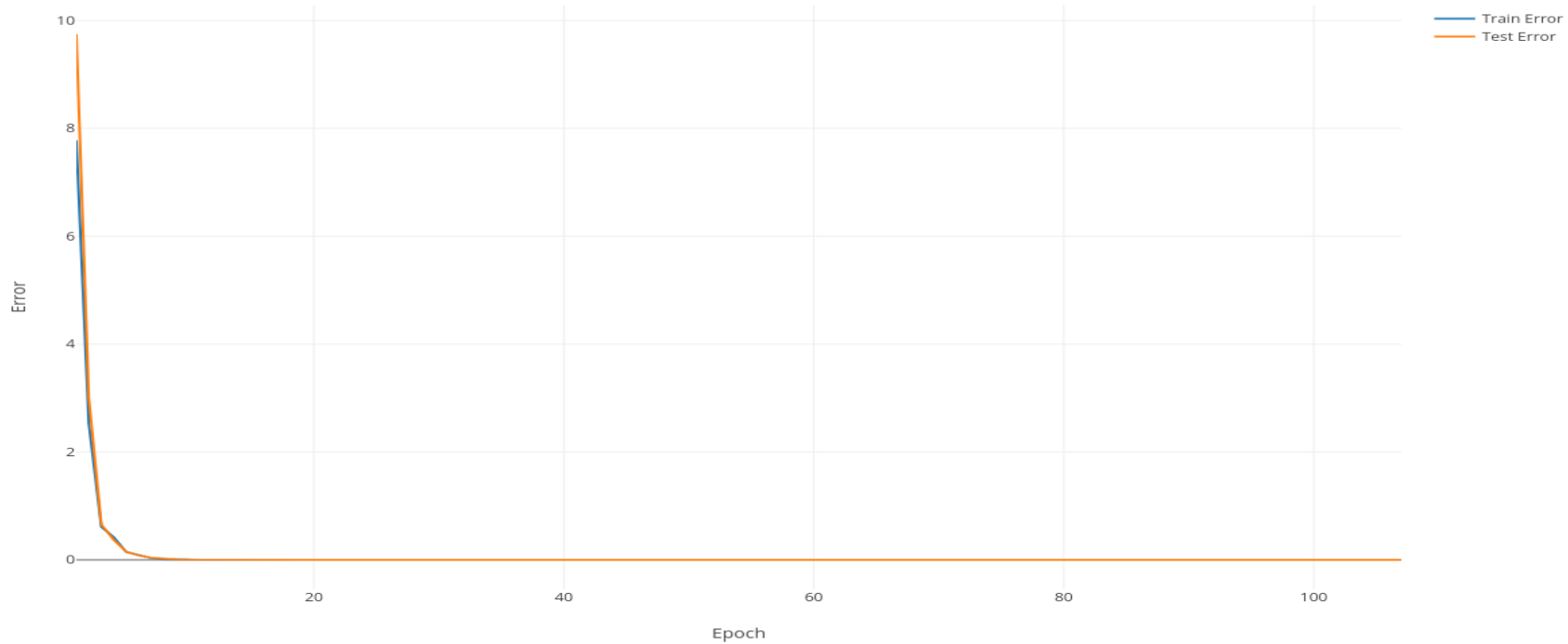
# Unit tests results:

SGD without momentum:



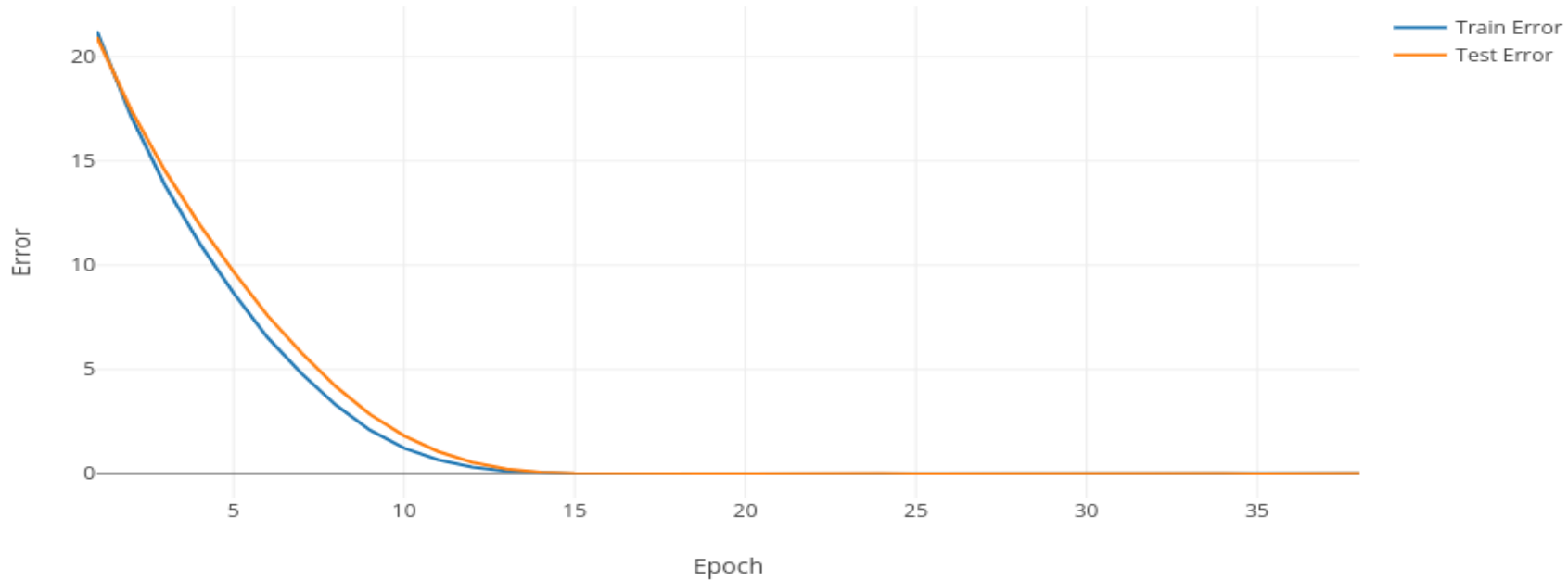
# Unit tests results:

SGD with momentum:



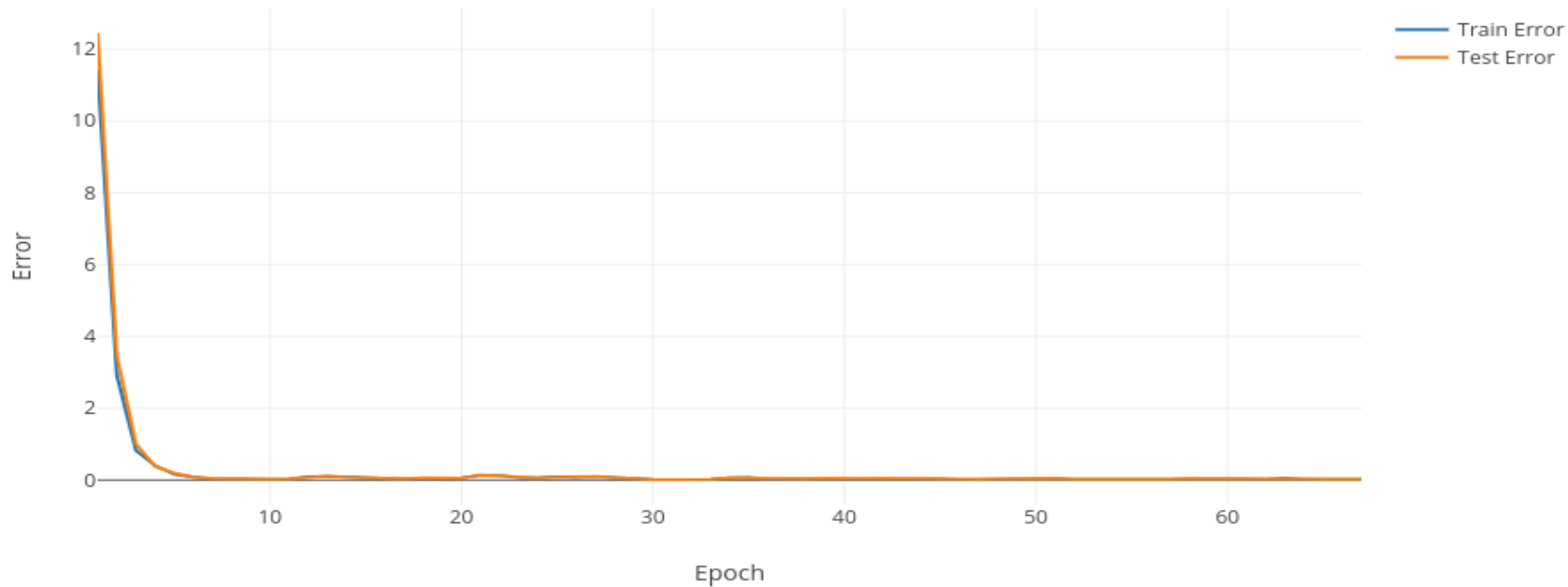
# Unit tests results:

RMSProp without momentum:



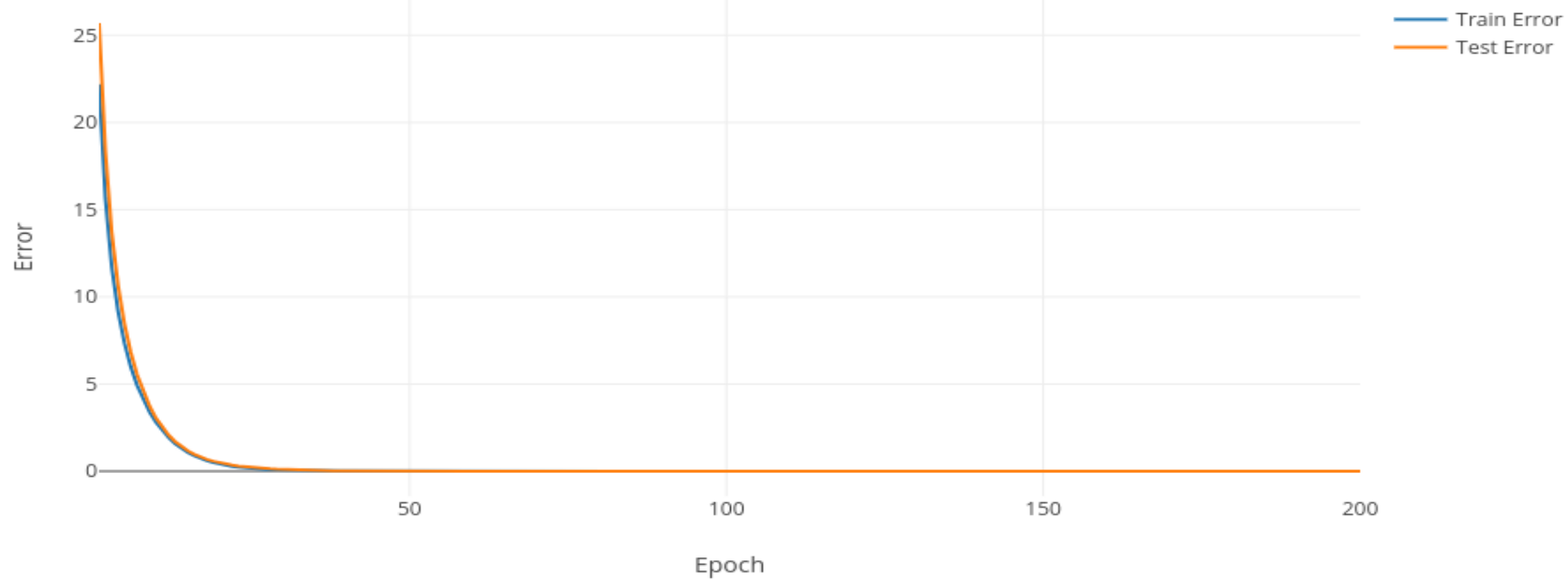
# Unit tests results:

RMSProp with momentum:



# Unit tests results:

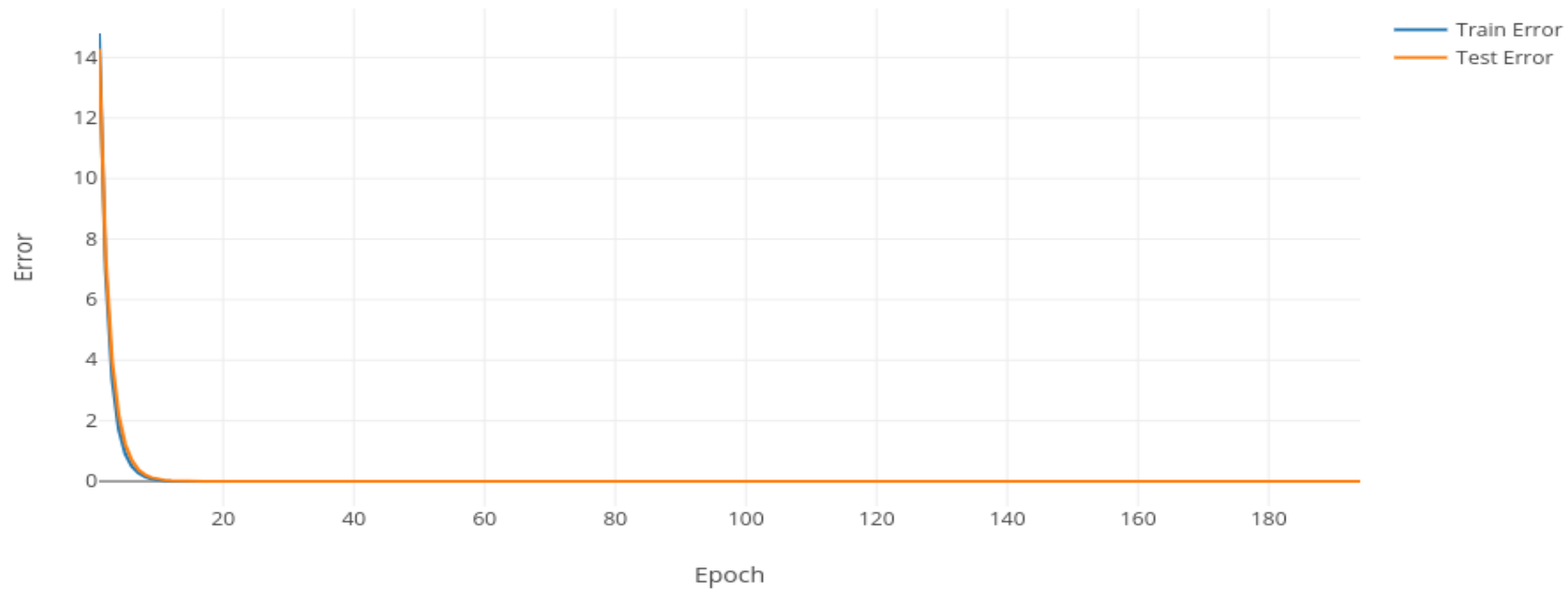
Adam:





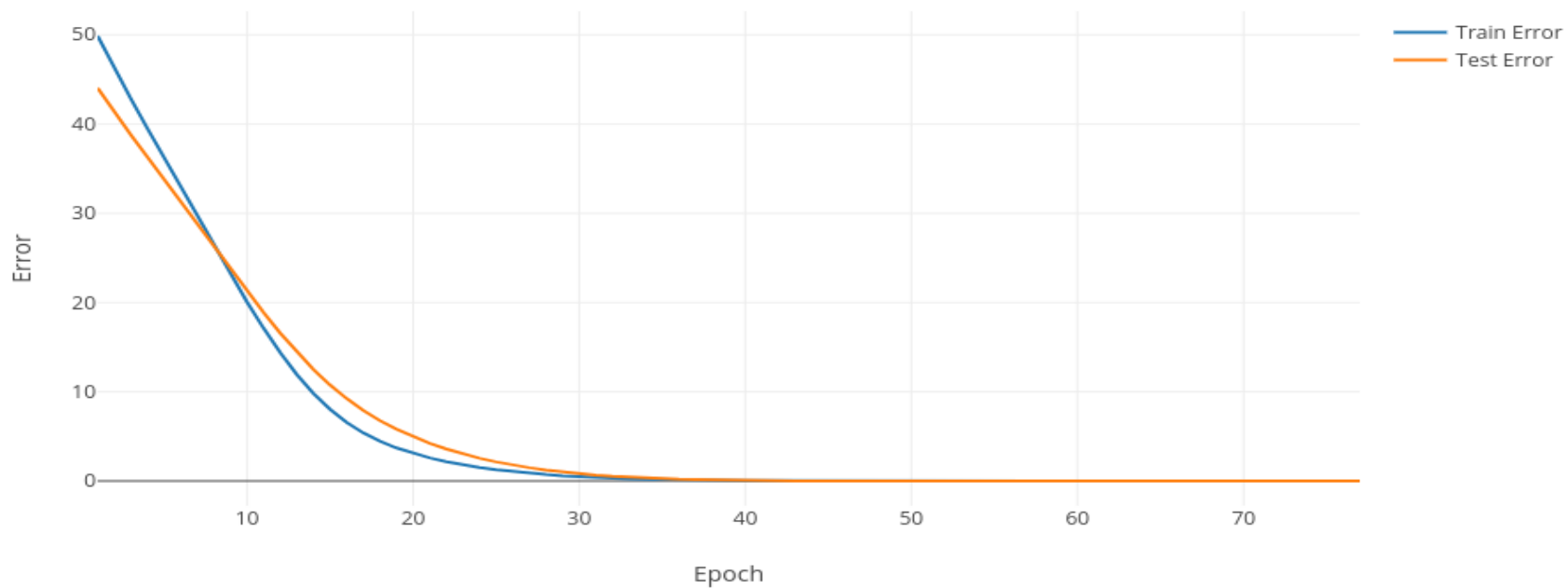
# Unit tests results:

Adagrad:



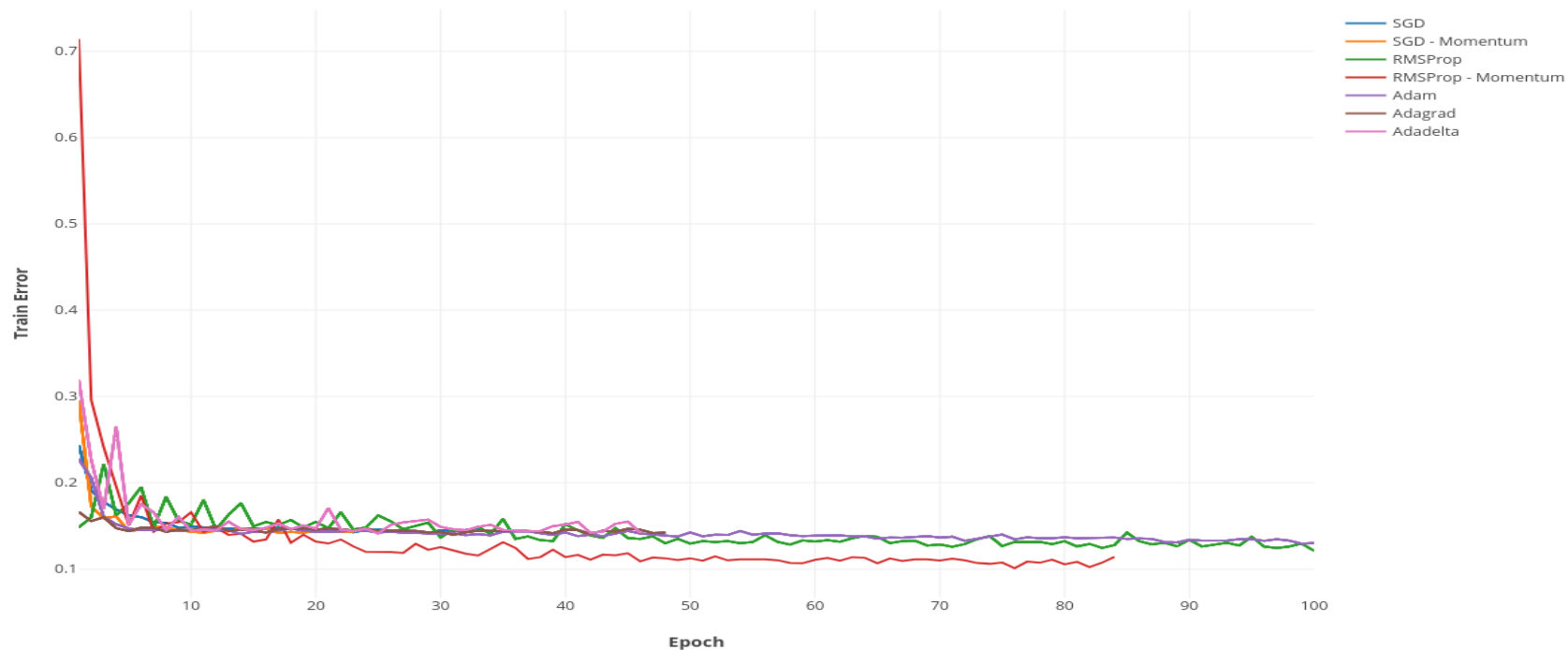
# Unit tests results:

Adadelta:



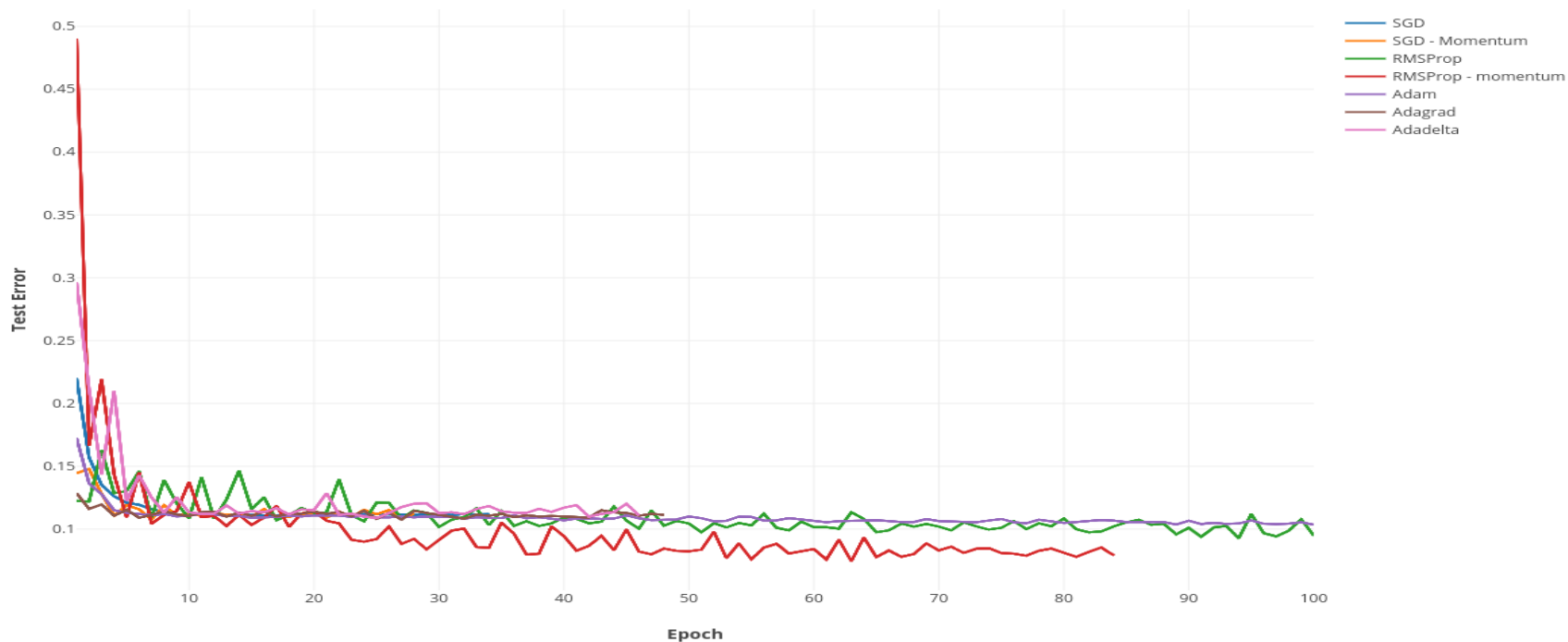
# MethodDL test results:

TMVA Optimizers Training Errors:



# MethodDL test results:

TMVA Optimizers Test Errors:



# MethodDL test results:

Optimizer	Start Epoch	Train Error	Test Error	End Epoch	Train Error	Test Error
SGD	1	0.243649	0.220027	34	0.141692	0.110918
SGD - Momentum	1	0.295173	0.144552	28	0.141734	0.107662
RMSProp	1	0.148158	0.122384	100	0.121107	0.0926917
RMSProp - Momentum	1	0.71423	0.490361	84	0.101079	0.0745212
Adam	1	0.227436	0.172377	100	0.12949	0.103475
Adagrad	1	0.31913	0.128628	48	0.141207	0.107541
Adadelata	1	0.166242	0.296319	46	0.139588	0.108797

# Future Work:

- Implement other optimizers like Adamax, Nadam and Nesterov accelerated SGD optimizers.
- Add Weight Decay implementation to optimizers.
- Benchmark the individual optimizers on separate datasets with tensorflow.

Thank  
you