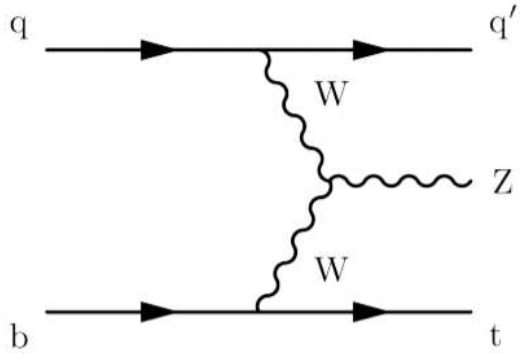
EVOLUTIONARY NEURAL NETWORK

CHELSEA JOHN
INTERNSHIP 2020
UNIVERSITY OF BONN

Contents

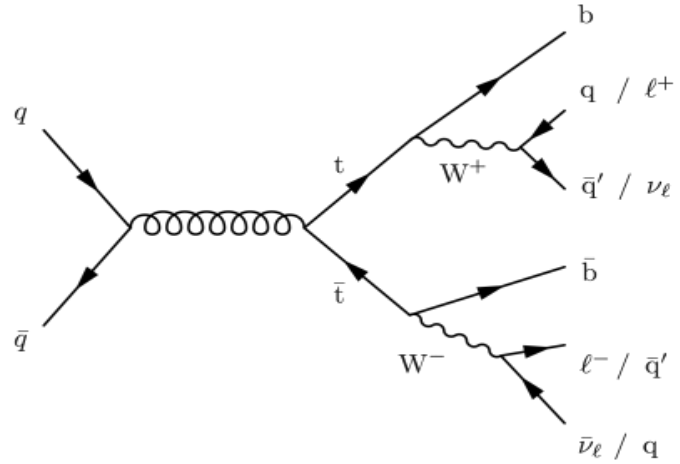
AIM: Separate Signal From Background Using Neural Networks

Signal : **tZq**



https://images.app.goo.gl/TgDpqtfyigJ8qg2ZA

Background : **ttbar**



https://wiki.physik.uzh.ch/cms/latex:feynman

sFile: /cephfs/user/s6taholm/tHq/run/12-10-hadhad/out_12-10-tZq.root

bFile : /cephfs/user/s6taholm/tHq/run/12-10-hadhad/ttbar.root

Variables Used For Distinguishing tZq From ttbar:

- eta_jf - Forward jet eta
- pt_jf - Forward jet transverse momentum
- mass_jf - Forward jet mass
- phi_jf - Forward jet phi
- eta_b - Eta of b
- pt_b - Transverse momentum of b
- phi_b - Phi of b
- HvisMass - Heavy Boson mass
- M_met - Mass of missing transverse energy
- Reco_w_mass_1 - Reconstructed W-1 mass
- Reco_w_mass_2 - Reconstructed W-2 mass

Grid Search Results

Deep Neural Network

NETWORK STRUCTURE	
No: of nodes	[10 - 100]
Hidden layers	[1 – 4]
Epochs	[50 - 250]
Dropout	[0 - 0.9]
Batchsize	[10 - 1000]
activations	ReLu, ReLu, Sigmoid
Initialization	Lecun_normal

OPTIMIZER	
Optimizer	Adam , SGD
Learning rate	SGD : [0.01 -0.99]
Decay	0
Momentum	SGD: [0 – 0.8]
Nesterov	True
Loss	Binary cross entropy

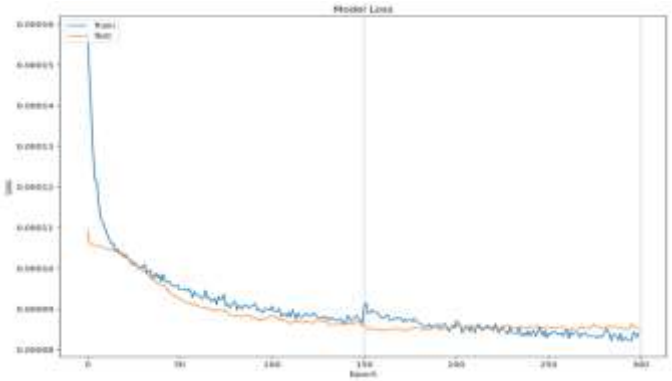
TRAINING SETUP	
K-fold splits	[1,2,3,4]
Features	conf/tZq_VarList_ckirfel.txt
Full Features	conf/tZq_VarList_ckirfel.txt
evFrac	0.25
UseBatchNorm	True
Estop	None

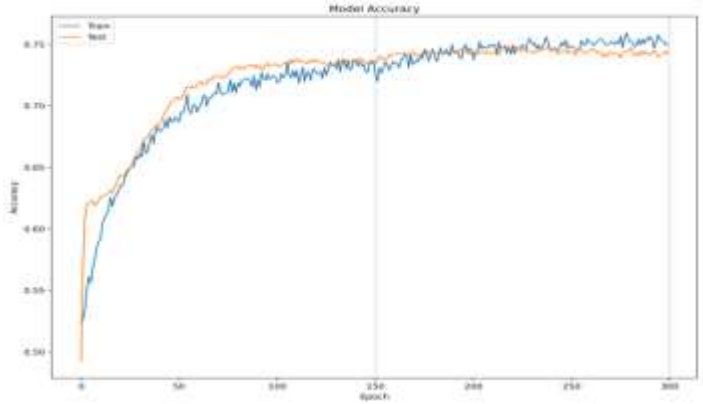
Graphs Analyzed

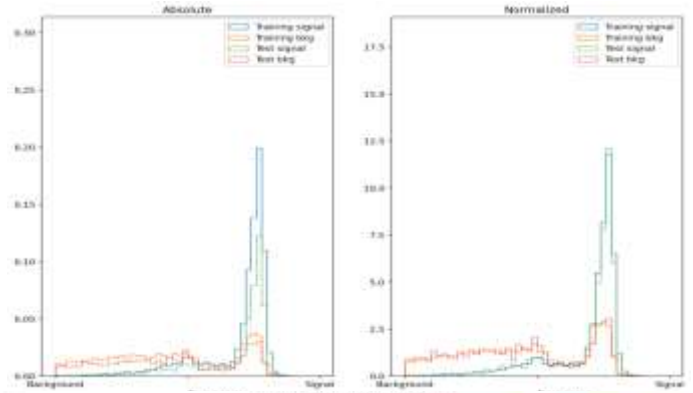
- Model Accuracy (Accuracy vs Epoch)
- Model loss (Loss vs Epoch)
- Response (Signal – 1, Background – 0)
- Receiver Operating Characteristic(ROC)

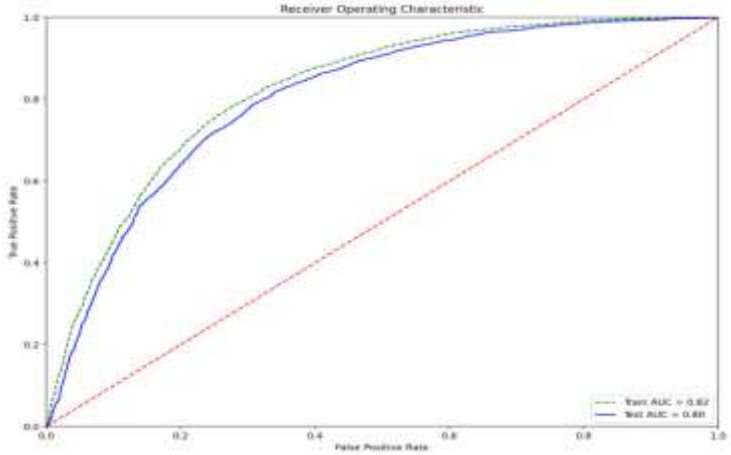
Optimizer: ADAM

Epoch [50,250]	Batch size [10,1000]	Hidden layers [1,4]	No of nodes [10,100]	K-fold splits [1,4]	Dropout [0,0.9]
150	200	2	50	2	0.8



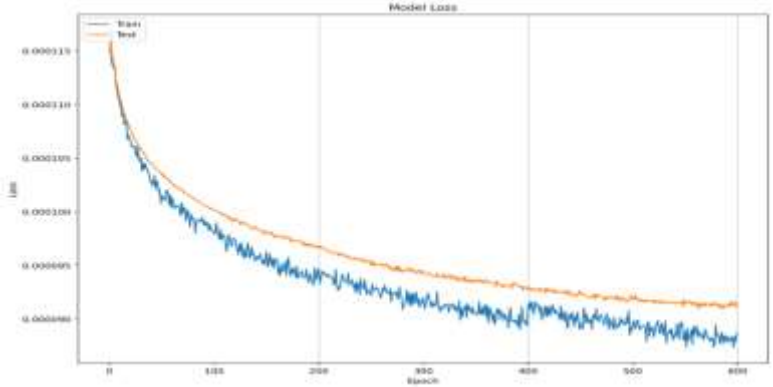


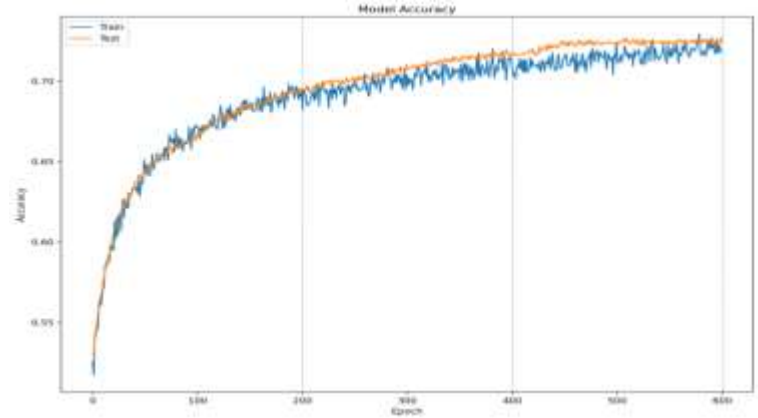


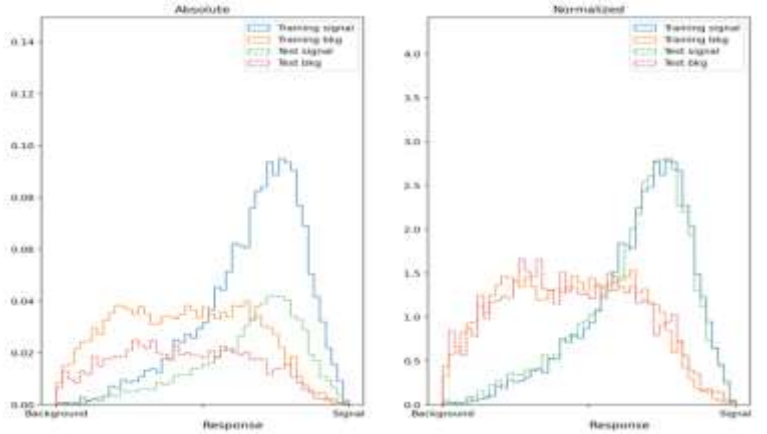


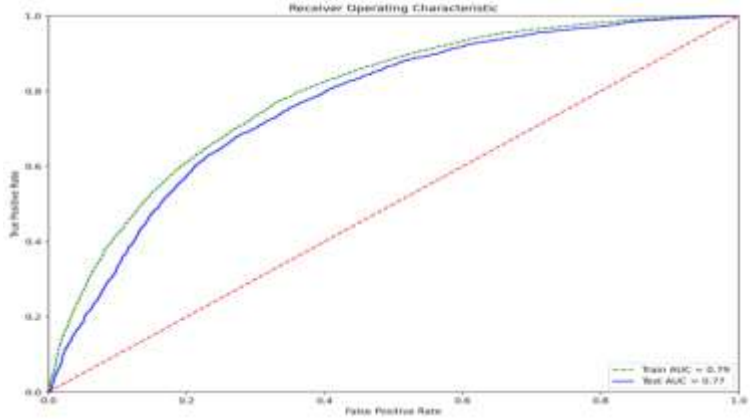
OPTIMIZER: SGD

Epoch [50,250]	Batch size [10,1000]	Hidden layers [1,4]	No of nodes[10,100]	K-fold [1,4]	Dropout [0,0.9]	Learning rate[0.01,0.99]	Momentum [0,0.8]
200	20	2	40	3	0	0.95	0.3









Evolutionary Neural Network

Evolutionary Neural Network

The general implementation is as follows

- Step One:
  - Generate the initial population of individuals/genotypes randomly. (First generation)
- Step Two: Repeat the following breeding steps until termination:
  - Evaluate the fitness of each genotype in the generation, i.e. here using the Loss, Accuracy, ROC and Response graphs
  - Select the fittest genotype for reproduction(Parents) [selecting the best genotype]
  - Breed new genotypes through crossover /mixing and mutation operations to give birth to offspring.
  - Replace the least-fit genotypes of the generation with new genotypes

VARIABLES

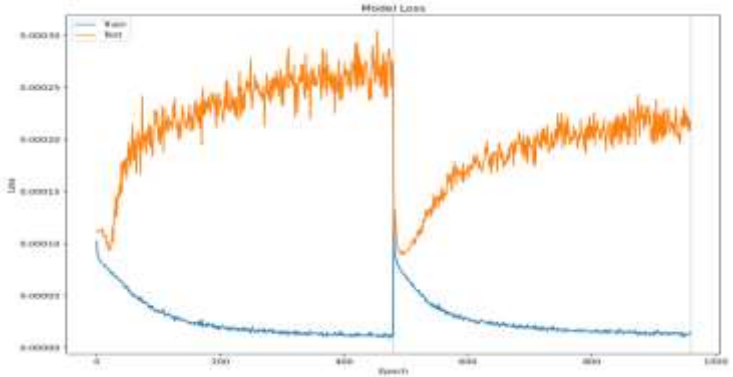
TRAINING SETUP	
Optimizer	Adam
Initialization	random_uniform
Nesterov	True
Loss	Binary cross entropy
Decay	0
Momentum	0.001
Features	conf/tZq_VarList_ckirfel.txt
Full Features	conf/tZq_VarList_ckirfel.txt
evFrac	0.25
UseBatchNorm	True
Estop	None

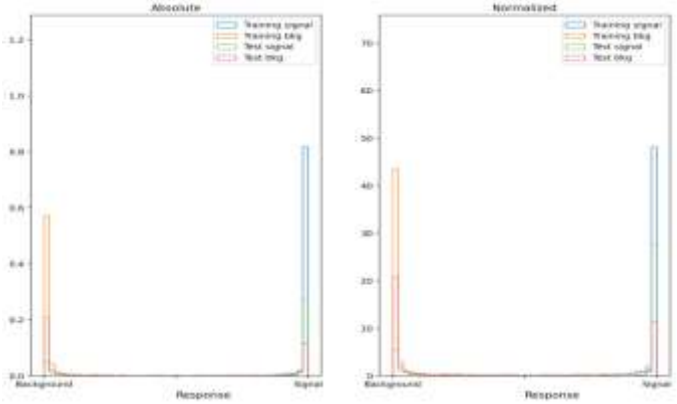
Hyper parameters

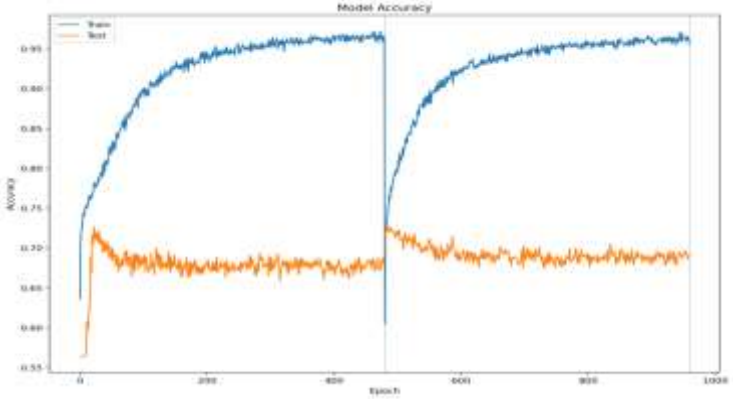
Hyper parameter	No: of Selections	Start	Stop	Step
Layers[]	3	1	10	1
Nodes[]	3	1	100	5
Dropout[]	3	0	1	0.10
Learning rate[]	2	0.001	0.1	0.001
Epoch[]	2	50	500	50
Batch Size[]	2	100	2000	10
Activation function[]	2	['relu', 'elu']		

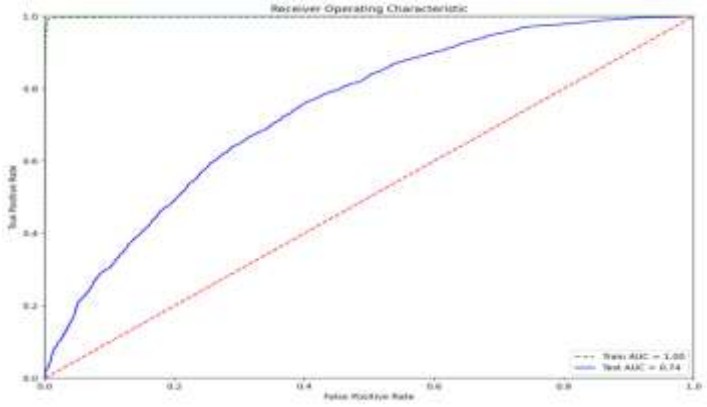
Selecting Best Genotype: with Max AUC

Gen	Mixing	Mutation	Epoch [50,500]	Batch [100,2000]	Layers [1,10]	Nodes [1,100]	K-fold splits	Dropout [0,1]	Activation [elu,relu]
3	5	5	480	440	7	91	2	0.1	Relu



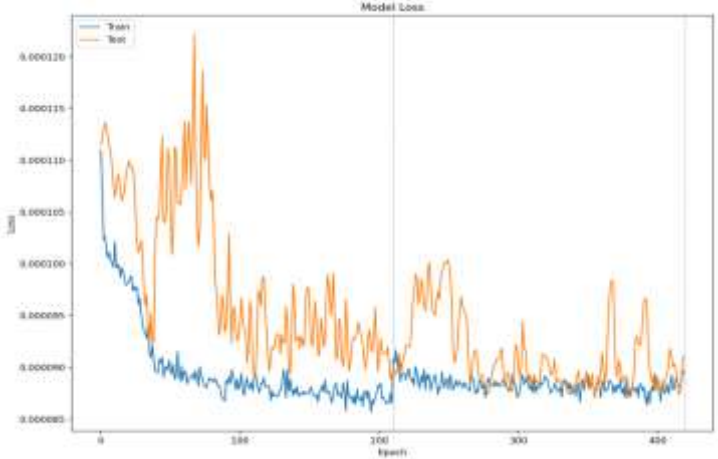


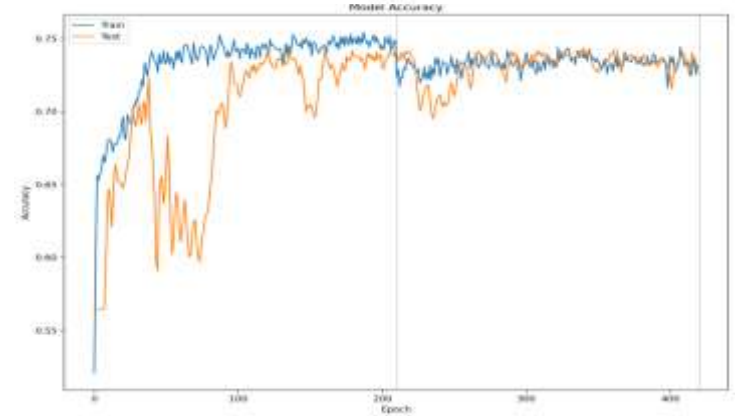


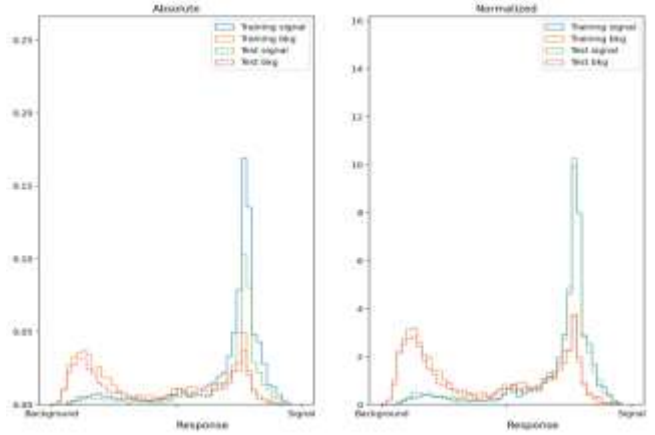


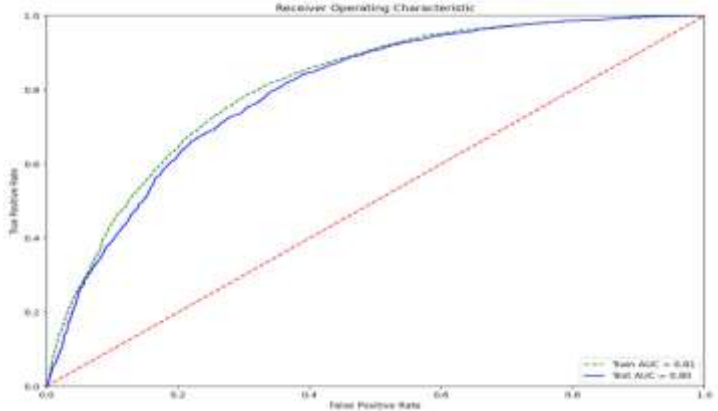
Selecting Best Genotype: with Least DeltaLoss

Gen	Mixing	Mutation	Epoch [50,500]	Batch [100,2000]	Layers [1,10]	Nodes [1,100]	K-fold splits	Dropout [0,1]	Activation [elu,relu]
3	5	5	210	1530	6	8	2	0.3	elu







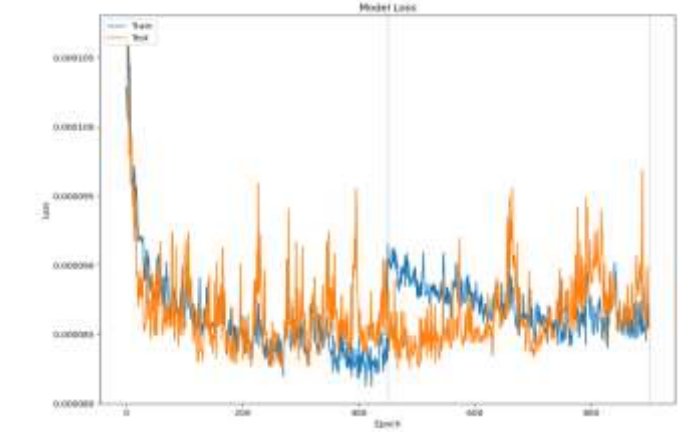


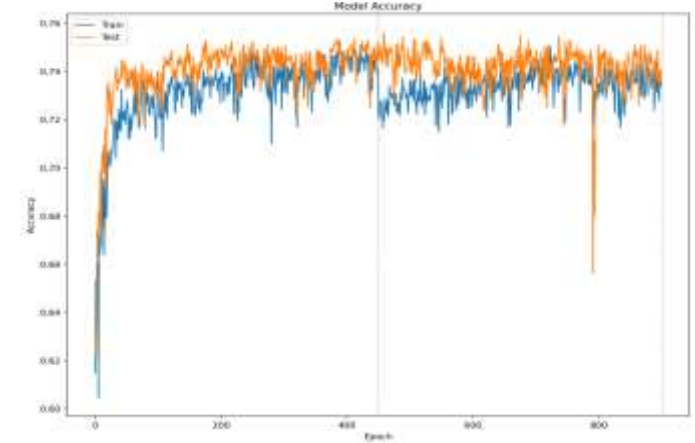
Selecting Best Genotype: Using Auc and DeltaLoss

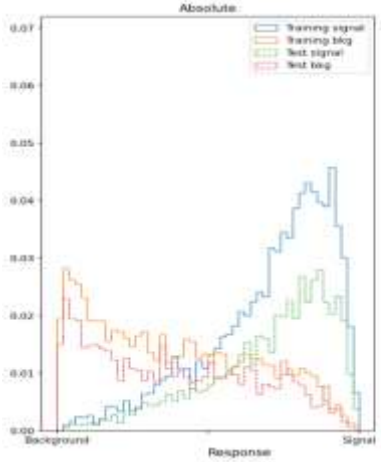
```
    # sorting wrt to max AUC and min deltaLoss
    genotypeSummary_1 = genotypeSummary.sort_values(by ='AUC', ascending = False)
    genotypeSummary_2 = genotypeSummary.sort_values(by = 'deltaLoss', ascending = True)
    test1_index = genotypeSummary_1.index
    test2_index = genotypeSummary_2.index
    best_index =[]
    print('AUC index: ',test1_index,'deltaLoss index: ',test2_index)
    for i in range(int(len(test2_index)/2)):
        if test1_index[i] == test2_index[i]:
            best_index.append(test2_index[i])
    if ( len(best_index) > 0 and len(best_index)< len(test2_index)):
        remaining_index = [ elem for elem in test2_index if elem not in best_index ]
        final_index = best_index + remaining_index
        print('final index:', final_index)
        genotypeSummary = pd.DataFrame(genotypeSummary, index = final_index)
    elif (len(best_index) == len(test2_index)):
        genotypeSummary = pd.DataFrame(genotypeSummary, index = best_index)
    else:
        genotypeSummary = genotypeSummary_2
```

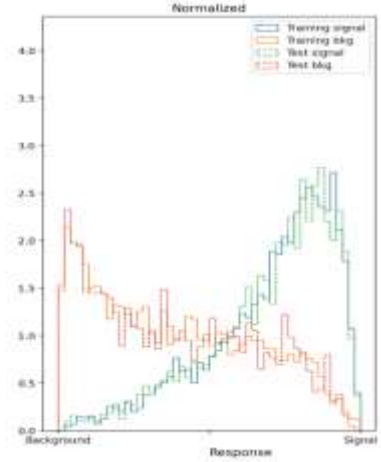
Selecting Best Genotype: Using AUC and DeltaLoss

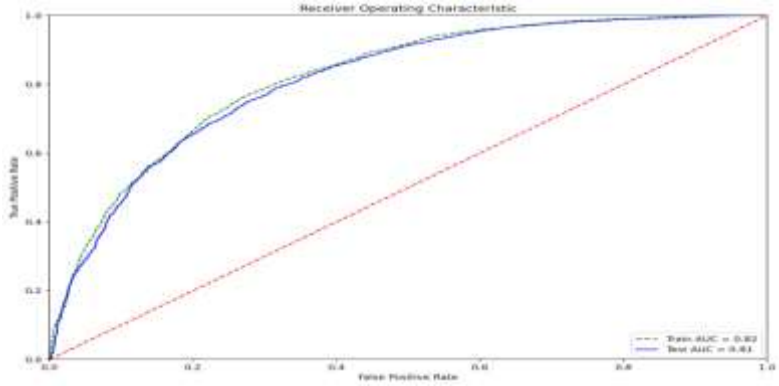
Gen	Mixing	Mutation	Epoch [50,500]	Batch [100,2000]	Layers [1,10]	Nodes [1,100]	K-fold splits	Dropout [0,1]	Activation [elu,relu]
3	5	5	450	780	1	59	2	0.6	elu





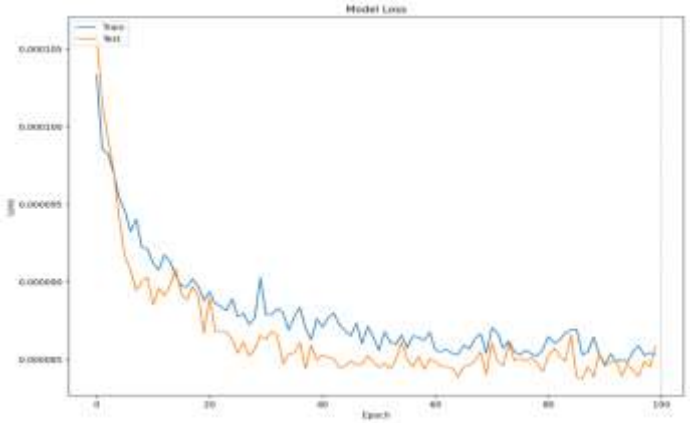


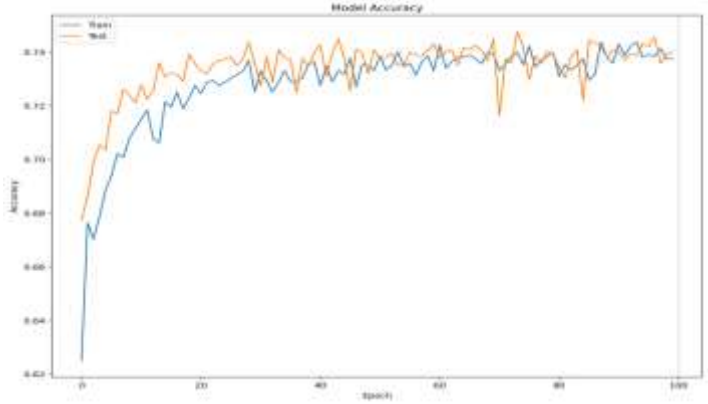


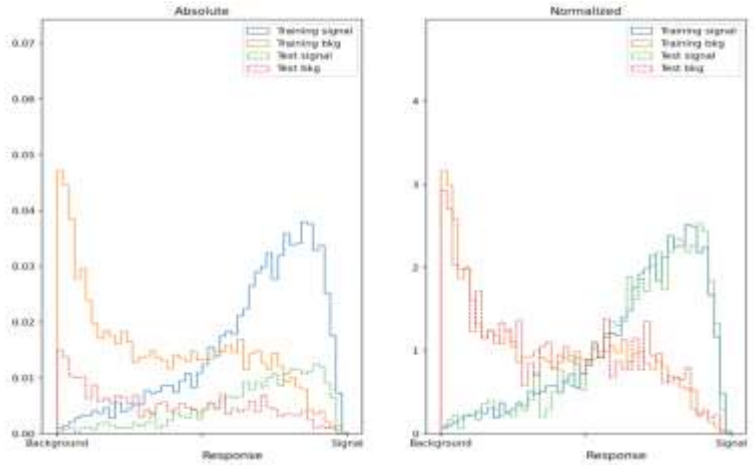


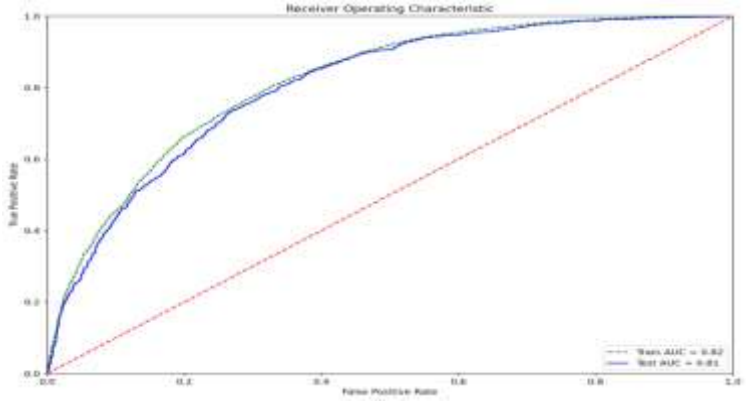
Fine Tuning the Best Genotype

Epoch	Batch	Layers	Nodes	K-fold splits	Dropout	Activation
100	780	1	59	1	0.6	elu





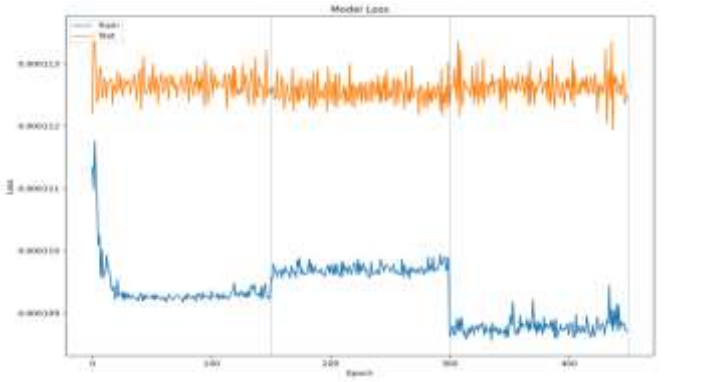


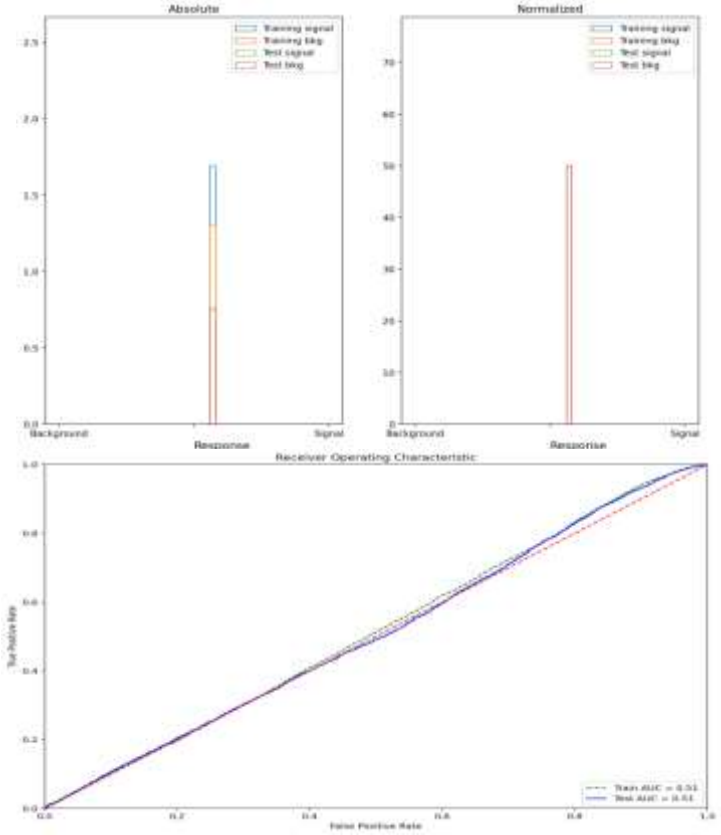


Verifying Best Genotype

Best:

Gen	Mixing	Mutation	Epoch	Batch	Layers	Nodes	K-fold splits	Dropout	Activation
3	3	3	150	920	5	16	3	0.9	elu





Verifying Best Genotype

- Check the mutation, mixing, and best results

Mixing:

Generation	layers	nodes	dropout	lr	epoch	batchsize	activation
1	5	16	0.9	0.02	150	920	['elu', 'sigmoid']
1	3	56	0.9	0.02	150	920	['elu', 'sigmoid']
1	3	16	0.9	0.01	150	920	['elu', 'sigmoid']
2	5	16	0.9	0.01	150	920	['elu', 'sigmoid']
2	4	17	0.9	0.02	150	920	['elu', 'sigmoid']
2	3	16	0.9	0.01	150	920	['elu', 'sigmoid']
3	4	14	0.9	0.01	150	920	['elu', 'sigmoid']
3	3	17	0.9	0.02	150	920	['elu', 'sigmoid']
3	5	16	0.9	0.04	150	920	['elu', 'sigmoid']
4	1	14	0.9	0.02	150	920	['elu', 'sigmoid']
4	3	16	0.9	0.02	150	920	['elu', 'sigmoid']
4	5	17	0.9	0.04	150	920	['elu', 'sigmoid']

Verifying Best Genotype

Mutation:

Generation	layers	nodes	dropout	lr	epoch	batchsize	activation
1	2	16	0.9	0.04	150	920	['elu', 'sigmoid']
1	4	55	0.9	0.01	150	920	['elu', 'sigmoid']
1	5	17	0.9	0.02	150	920	['elu', 'sigmoid']
2	1	14	0.9	0.04	150	920	['elu', 'sigmoid']
2	3	14	0.9	0.03	150	920	['elu', 'sigmoid']
2	5	17	0.9	0.01	150	920	['elu', 'sigmoid']
3	3	12	0.9	0.04	150	920	['elu', 'sigmoid']
3	2	17	0.9	0.03	150	920	['elu', 'sigmoid']
3	3	16	0.9	0.01	150	920	['elu', 'sigmoid']
4	1	16	0.9	0.02	150	920	['elu', 'sigmoid']
4	1	14	0.9	0.04	150	920	['elu', 'sigmoid']
4	4	14	0.9	0.01	150	920	['elu', 'sigmoid']

SOLUTIONS :

- Increasing the number of generations and genotypes in mutations and mixing
- Broaden the value range of parameters to give sufficient variety in mixing and mutation combinations
- Add early stopping

Conclusion:

- Evolutionary Neural Network can produce results faster than Grid Search
- Epoch, Batch size and Activation Function are added as input variables to the Evolutionary Neural Network(ENN)
- ENN results are improved by selecting Best Genotype using both AUC and deltaLoss values
- The genotypes created by mixing and mutation as well as the best genotype of each generation is written to a text file for later reference.
- ENN results can be improved further by increasing the number and phase space of genotypes created.
- It would be interesting to add EarlyStopping to ENN

Reference:

Alejandro Martín, Raúl Lara-Cabrera, Félix Fuentes-Hurtado, Valery Naranjo, David Camacho,
EvoDeep: A new evolutionary approach for automatic Deep Neural Networks parametrisation,
Journal of Parallel and Distributed Computing,
https://www.sciencedirect.com/science/article/abs/pii/S0743731517302605