Identifying objects in ATLAS through machine learning techniques

By

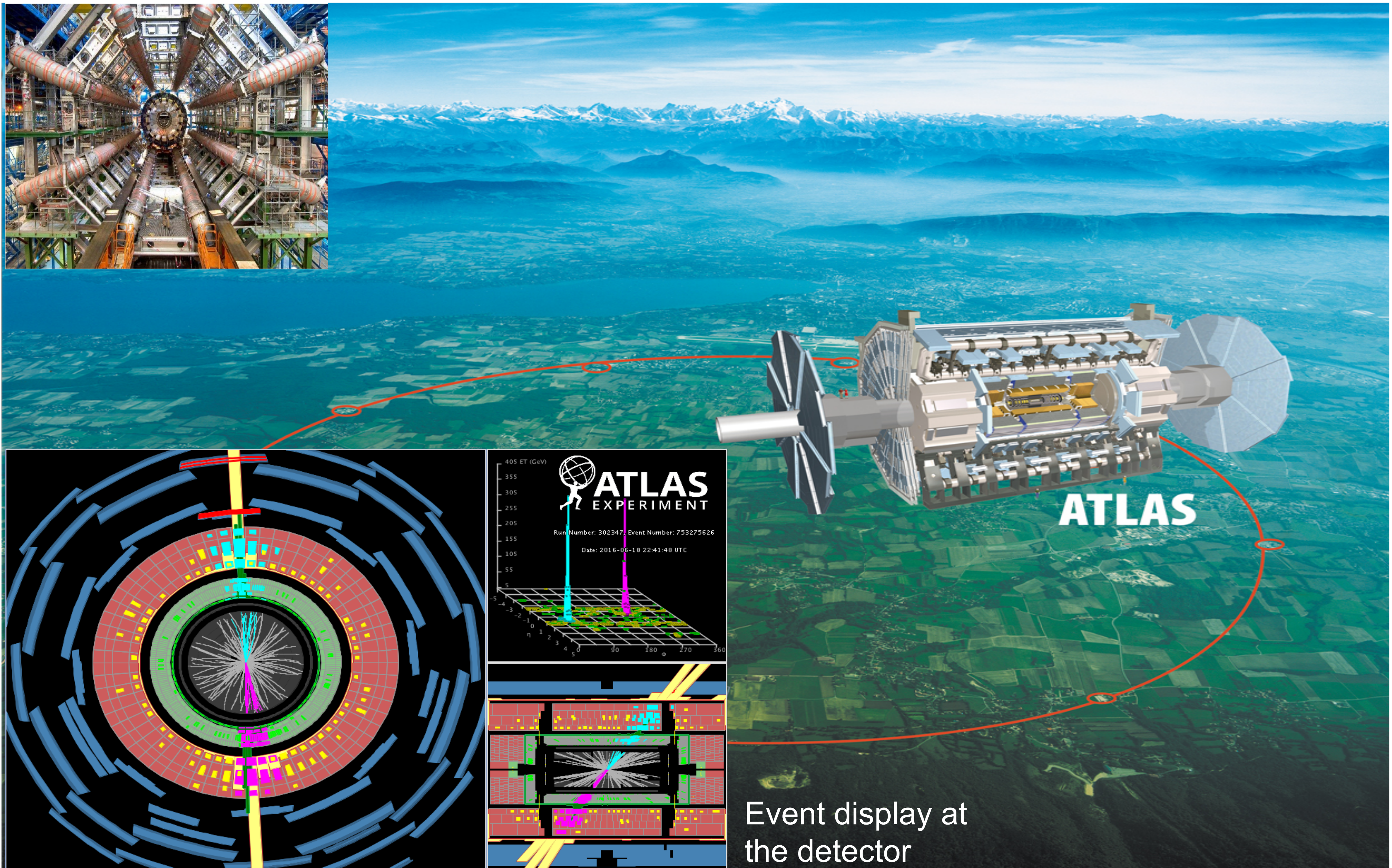
Wasikul Islam*, Nesar Soorve Ramachandra

*Research Assistant, Oklahoma State University
Research Aide, Argonne National Laboratory

Under the supervision of
Dr. J. Taylor Childers

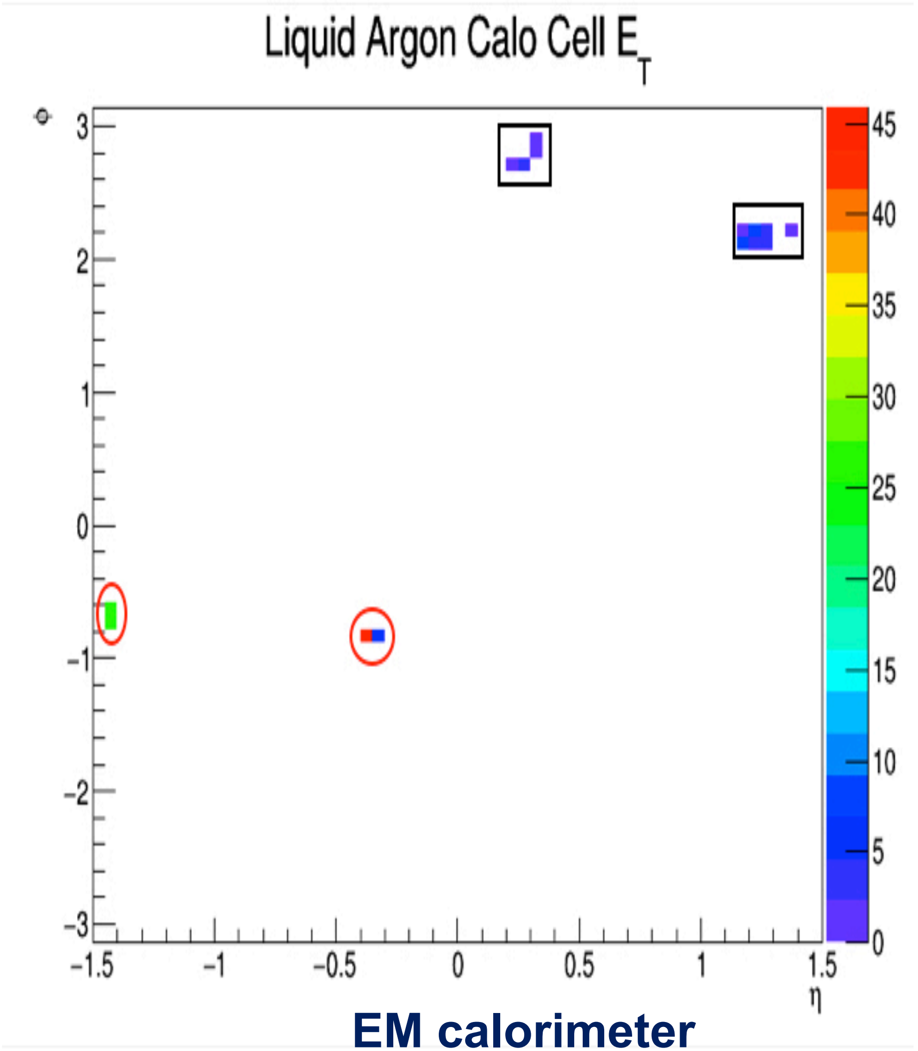
Young Scientists Symposium, 07.18.2017

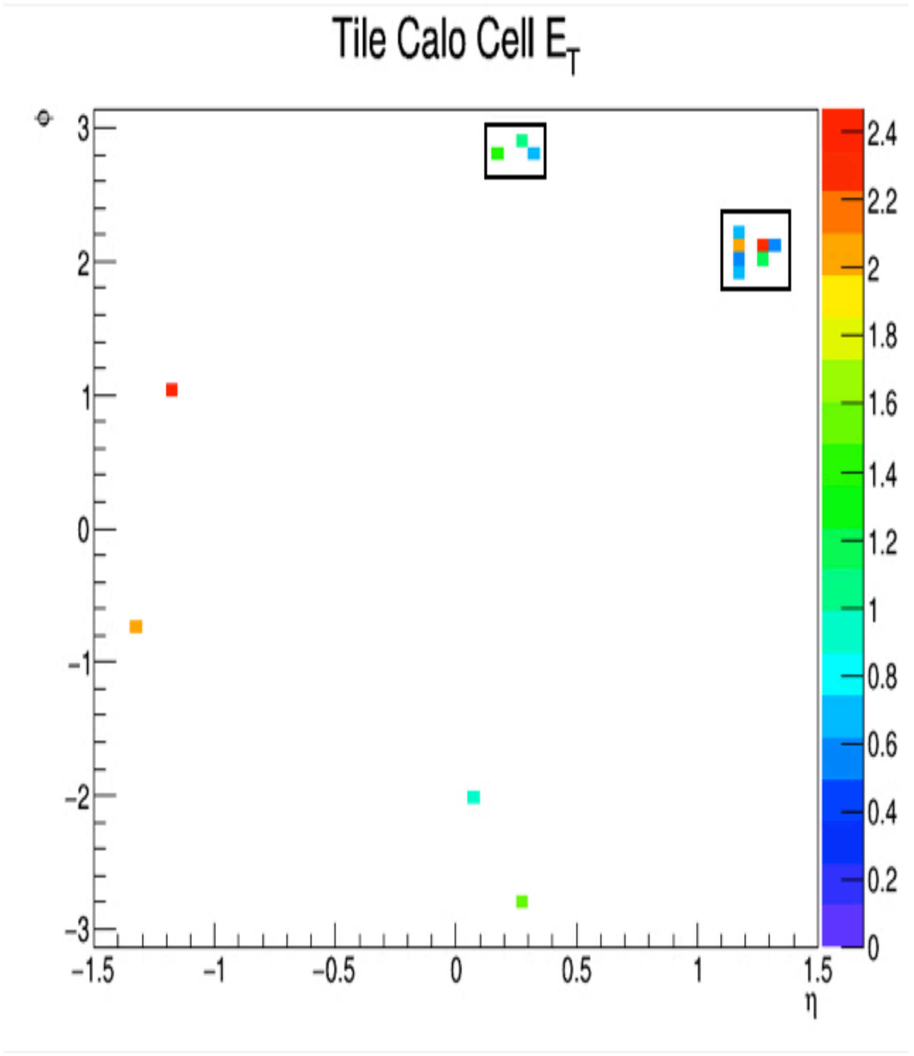
The ATLAS detector at CERN



Event display at
the detector

Imaging the hits in the calorimeter



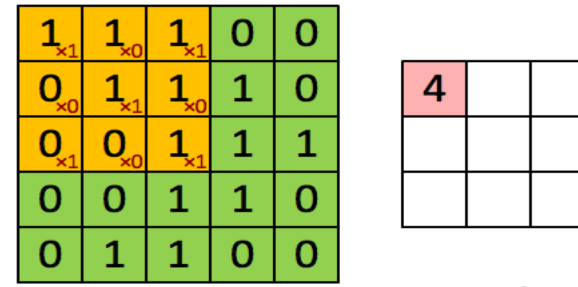


❖ Red circles are denoting electrons & Black rectangles are the jets at calorimeters above.

Strategy for using Machine learning techniques for object identification

- Firstly we dumped the object information from Zee, Zmumu, Ztautau events for the leptons and jet into subimages.
- Then we constructed a model of 2D Convolutional neural network following a popular 'Image classification model' of Cifar10.
- When we succeeded to have a working model, we extended our model for classifying objects of 4 classes I.e; Electrons, Muons, Tau leptons and Jets.
- Then we started playing with hyper parameters (Training rate, decay rate, batch size, number of epochs, loss functions, optimizers etc.) to optimize the accuracy of the model.

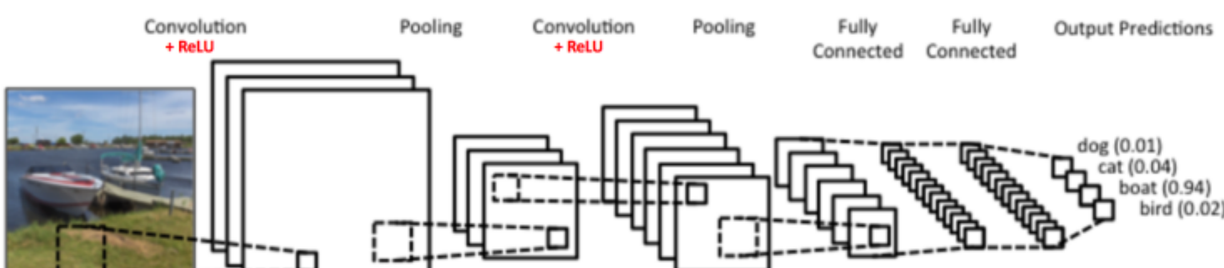
How our model of Convolutional neural network works



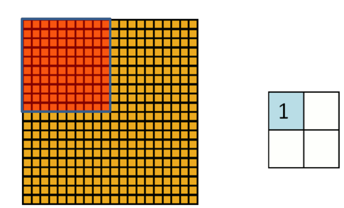
Image

Convolved Feature

- Our model is quite similar to the popular Cifar10 model.
- Our model has two channels : E.M & Hadronic.
- We have some layers of specific window sizes.
- We are also having different operations like MaxPooling, Flattening, Densing, Drop out etc.
- Our model uses ‘loss functions’ for the model while running.
- Also we it uses (different) ‘Optimizers’ to minimize the ‘loss function’ and to optimize the output of the model.
- And we have divided all the sub images into ‘Training data’ & ‘Testing data’ and within training data, used 20 percent of it for internally validating while it trains the model.



An example how image classification works using CNN

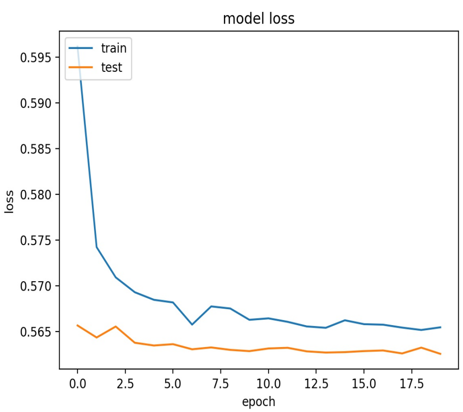


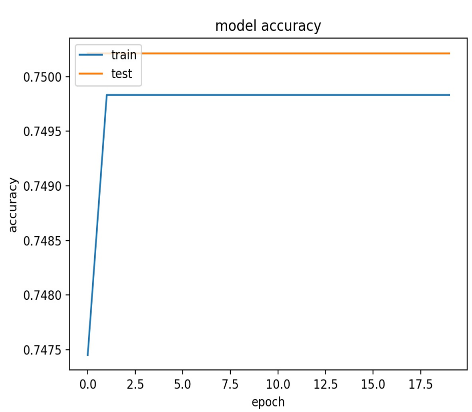
Convolved feature

Pooled feature

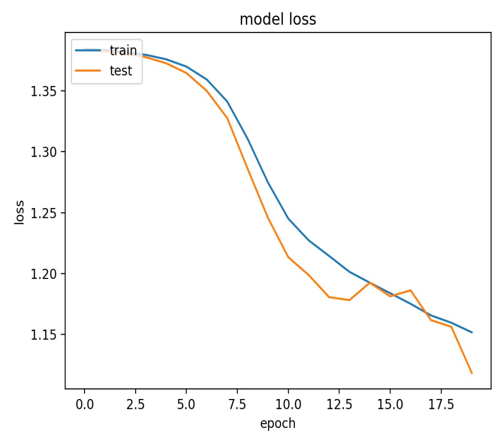
Some plots of ‘model accuracy’ and ‘loss’

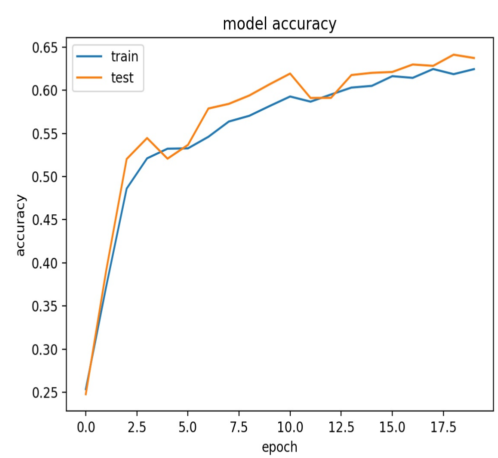
When we first started while have issues with proper mapping :



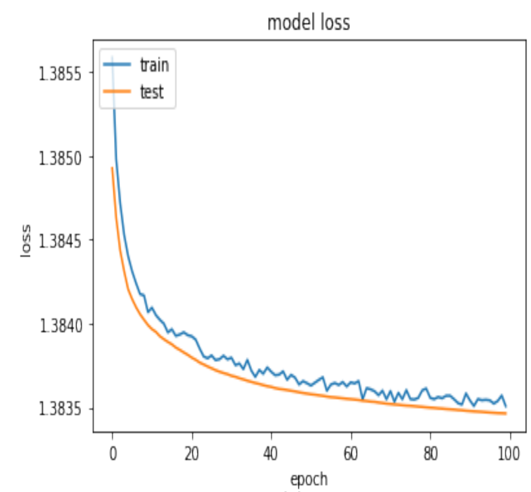


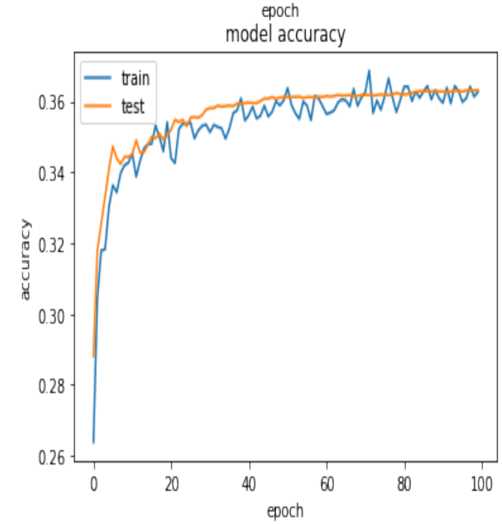
Some improvements while playing with parameters :



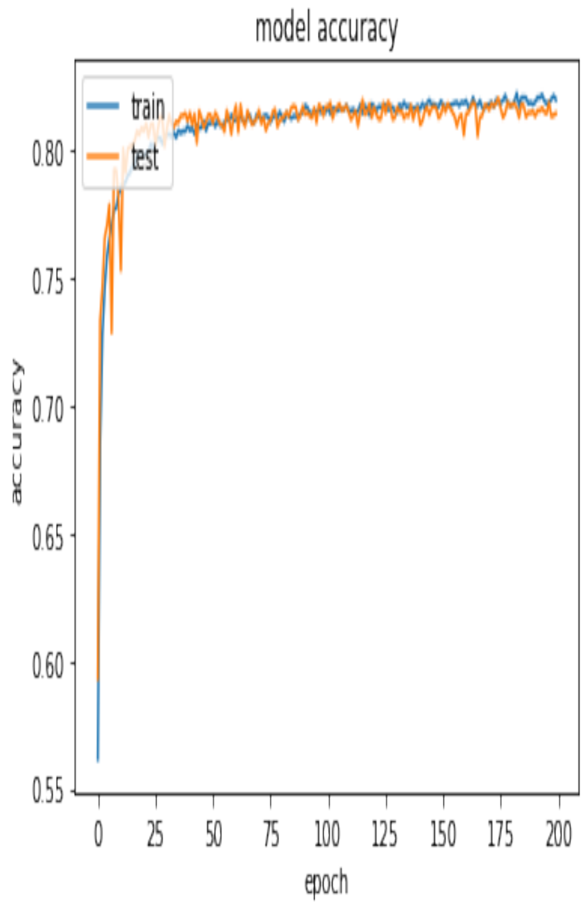


Results became better with more number of epochs

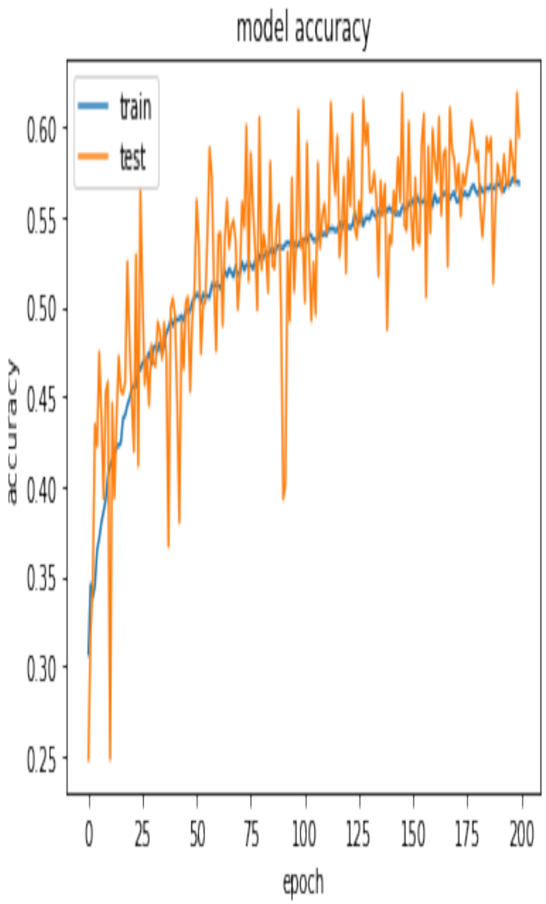




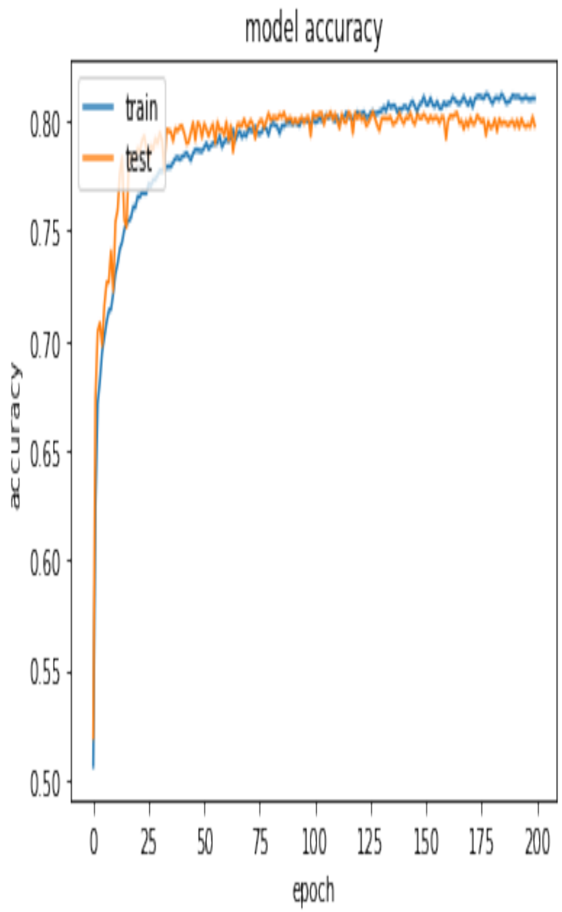
Varying the different ‘optimizers’ for model accuracy



Optimizer :RMSprop

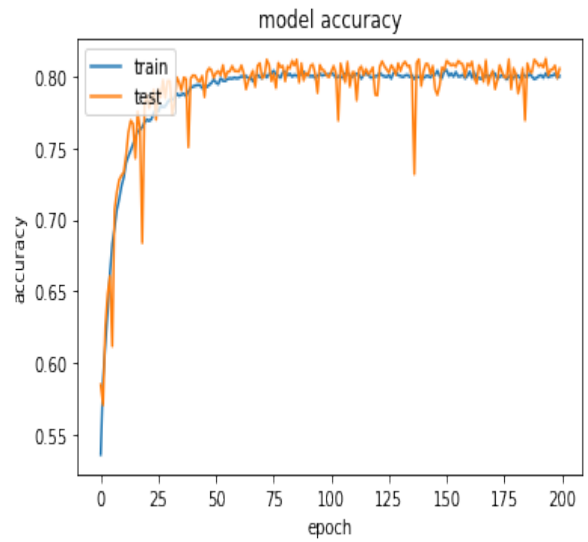


Optimizer : SGD

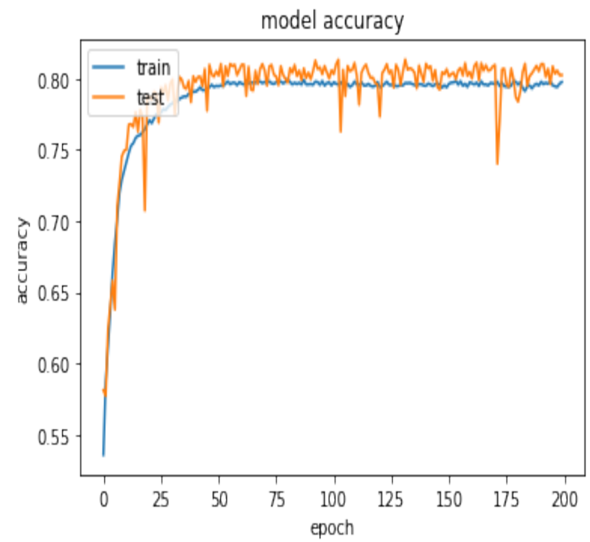


Optimizer : Adam

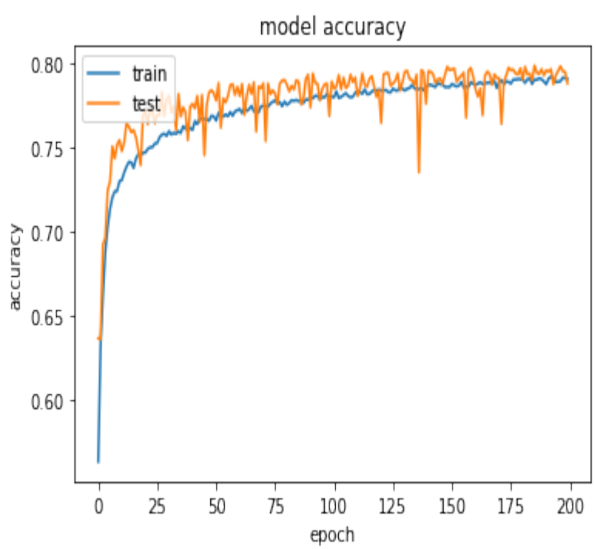
Experimenting with layers



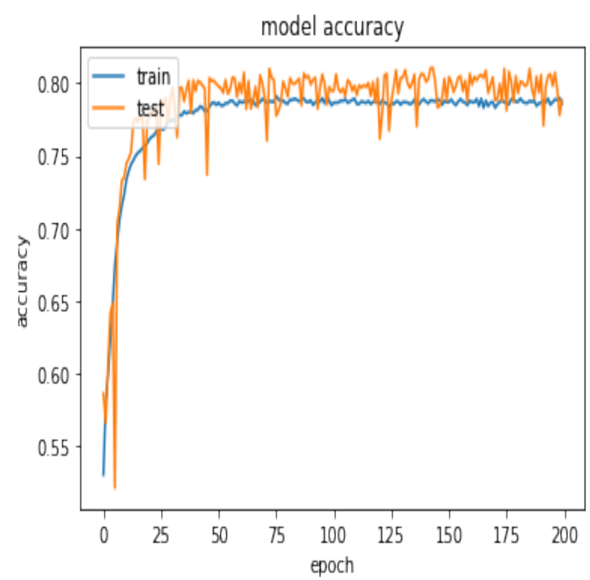
Adding (Conv2D(128, (3, 3)))



(Conv2D(32, (3, 3)))-> (Conv2D(64, (3, 3)))
Dropout : 0.5, 0.25, 0.5



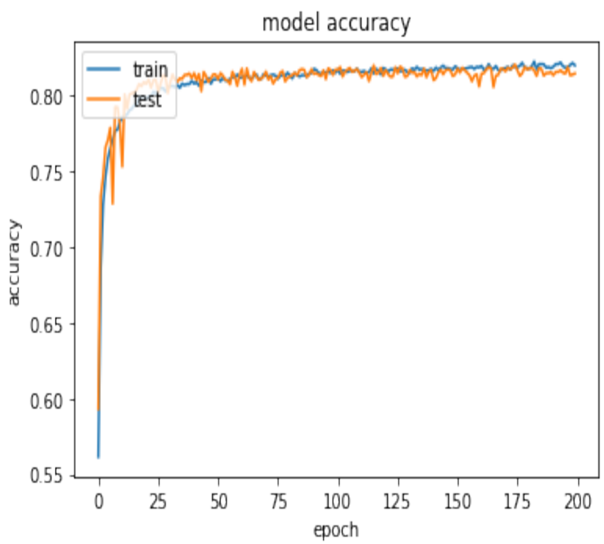
Activation (relu) -> Activation(selu)



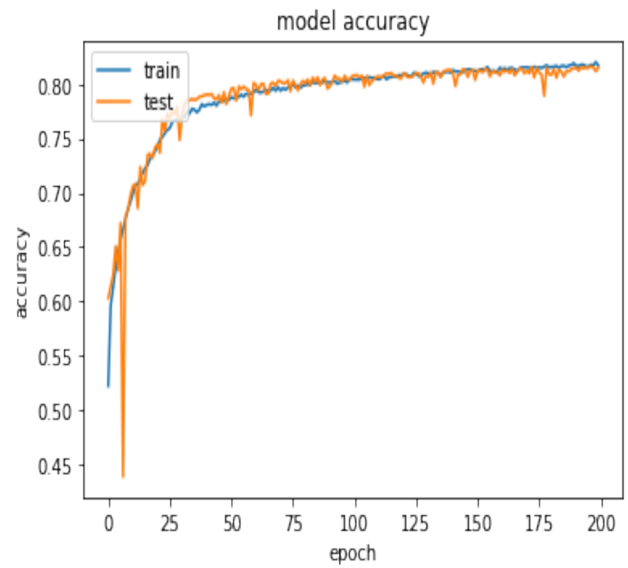
Activation (selu) -> Activation(relu)
Dropout : 0.5, 0.5, 0.5

Varying the different 'loss functions' and 'number of epochs' for model accuracy

For 200 Epochs :

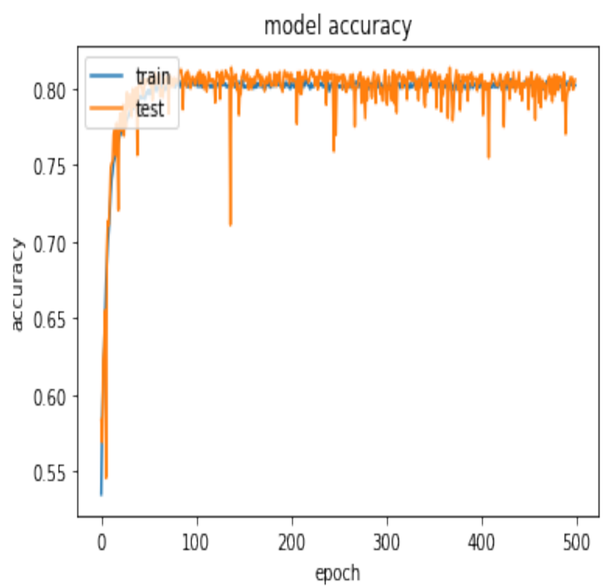


Loss function : Categorical Cross entropy

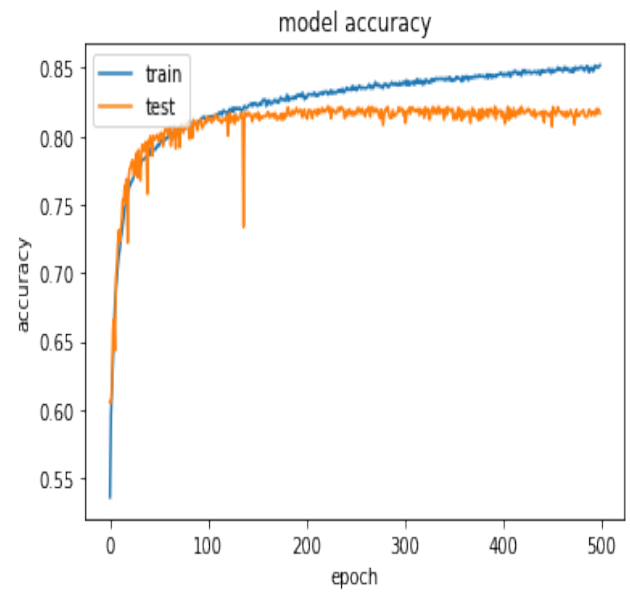


Loss function : Mean squared Error

For 500 Epochs :



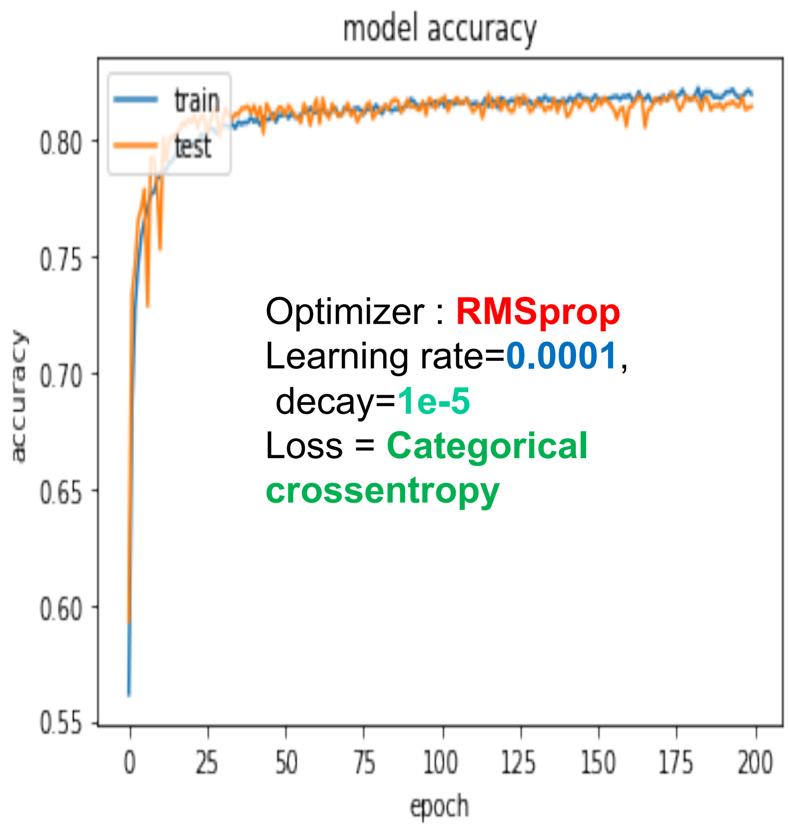
Loss function : Categorical Cross entropy



Loss function : Mean squared Error

Some ‘testing’ results from our best model

Our best model in terms of accuracy has the following parameters :



Optimizer : **RMSprop**
Learning rate=**0.0001**,
decay=**1e-5**
Loss = **Categorical crossentropy**

Testing electrons

True label : Electron

class 0 (electron)	class 1(jet)	class2(muon)	class3(tau lepton)
98.5014975	0.001655	0.0005293	1.4963153

Testing Jets

True label : Jet

class 0 (electron)	class 1(jet)	class2(muon)	class3(tau lepton)
0.0004065	**99.891948**	0.00014309	0.010750

Testing Muons

True label : Muon

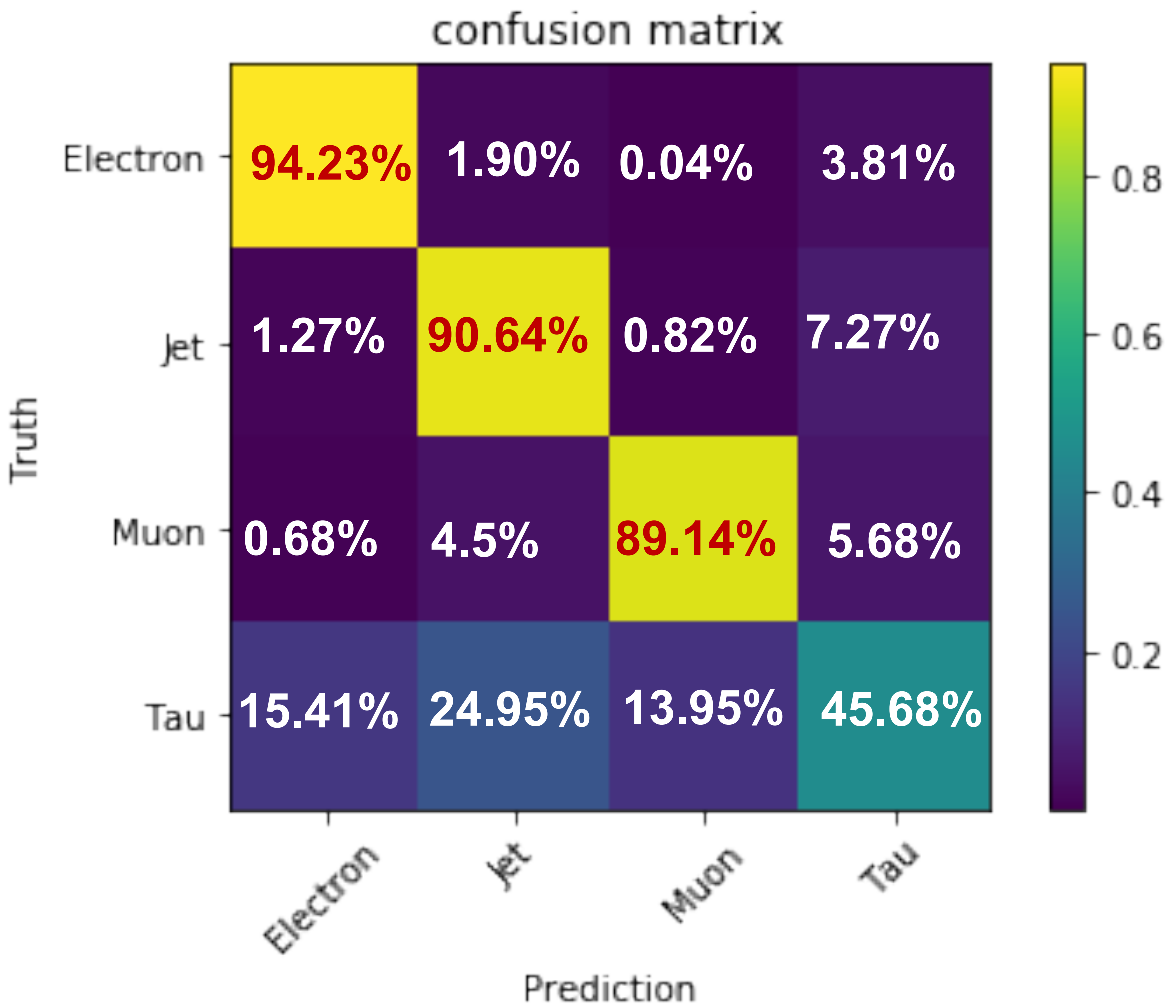
class 0 (electron)	class 1(jet)	class2(muon)	class3(tau lepton)
.0001563	.0032537	**95.1616049**	4.834976

Testing Tau leptons

True label : Tau lepton

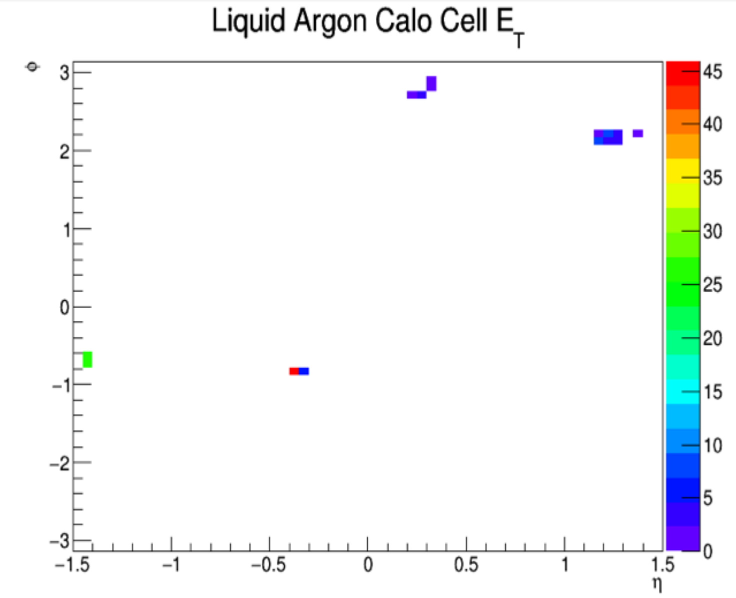
class 0 (electron)	class 1(jet)	class2(muon)	class3(tau lepton)
0.000590075	6.41879588	0.00014436	**93.5804665**

Confusion Matrix from the same model



Plans ahead…

- Our next plan is to apply this same machine learning techniques to identify "events" instead of identifying single object!



- And after that our plan will be to move to 3 dimensional machine learning of the detector and to modify our code for more complex cases of event reconstruction.


Thank You!

Backup slides

Optimization:

- Finding (one or more) minimizer of a function subject to constraints
- Most of the machine learning problems are, in the end, optimization problems.

Loss function: Categorical cross entropy :

For discrete p and q this means

$$H(p,q) = -\sum_x p(x) \log q(x).$$

Loss function: Mean Squared error:

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(\hat{Y}_i - Y_i)^2$$

Stochastic gradient descent

This method performs a parameter update for **each** training example $x^{(i)}$ and label $y^{(i)}$.

Update equation

$$\theta = \theta - \eta * \nabla_\theta J(\theta; x^{(i)}; y^{(i)})$$

We need to calculate the gradients for the whole dataset to perform **just one update.**

Advantage

It is usually **much faster** than batch gradient descent.

It can be **used to learn online.**

Disadvantages

It performs frequent updates with a **high variance** that cause the objective function to fluctuate heavily.

Ruder, Sebastian. "An overview of gradient descent optimization algorithms." *arXiv preprint arXiv:1609.04747* (2016).

RMSprop

RMSprop as well divides the learning rate by an exponentially decaying average of squared gradients.

RMSprop

$$E[g^2]_t = 0.9E[g^2]_{t-1} + 0.1g_t^2$$
$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{E[g^2]_t + \epsilon}} g_t$$

Ruder, Sebastian. "An overview of gradient descent optimization algorithms." *arXiv preprint arXiv:1609.04747* (2016).

Adam

Adam's feature :

- Storing an exponentially decaying average of past squared gradients v_t like Adadelta and RMSprop
- Keeping an exponentially decaying average of past gradients m_t, similar to momentum.

Counteracting these biases in Adam

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t}$$

$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t}$$

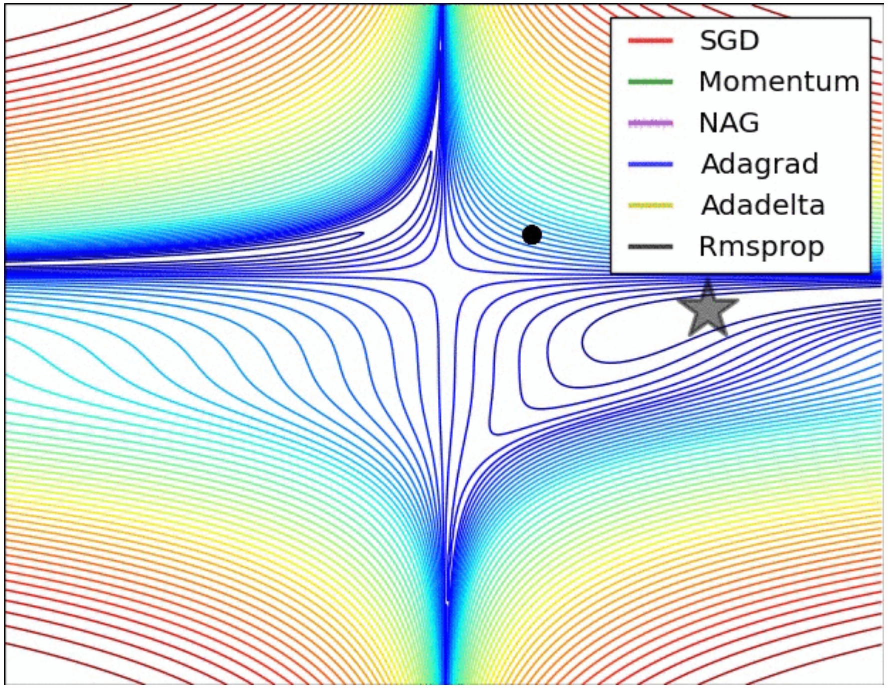
Adam

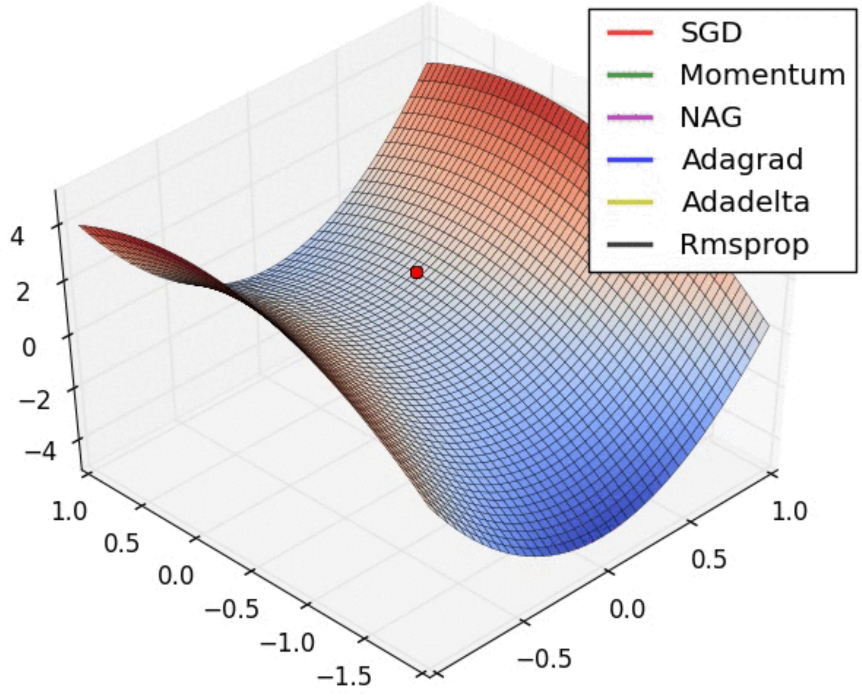
$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{v}_t} + \epsilon} \hat{m}_t$$

Ruder, Sebastian. "An overview of gradient descent optimization algorithms." *arXiv preprint arXiv:1609.04747* (2016).

Visualization of algorithms

As we can see, Adagrad, Adadelta, RMSprop, and Adam are most suitable and provide the best convergence for these scenarios.





Ruder, Sebastian. "An overview of gradient descent optimization algorithms." *arXiv preprint arXiv:1609.04747* (2016).