Development of a machine learning algorithm for continuous classification of the Higgs boson CP in decay $H \to \tau\tau$
Opracowanie algorytmu uczenia maszynowego dla ciagłej klasyfikacji stanu CP bozonu Higgsa w rozpadzie $H \to \tau\tau$

Sadowski Michał

July 15, 2019

1 Introduction

In recent years Machine Learning became very popular in solving classification problems. One of the fields in which it can be applied is solving High Energy Physics problems.

According to standard model Higgs boson is an elementary particle. One of the characteristic of Higgs boson is its CP state [4]. Machine Learning application in the classification of Higgs boson as scalar or pseudoscalar, looking at the features in kinematical distributions was investigated in paper [5]. In this project we examine methods of modification of this approach to allow continues classification for different mixing CP angles.

2 Input features

In this project I train and test my model using data generated for Higgs boson decay $H \to \tau\tau$ in $\rho\rho$ channel. The feature set used as an input of neural network consisted of 4-vectors of all decay products, including neutrinos. As described in [1] to help classifier perceive the dependency between these values and labels, particles were rotated and boosted to the rest frame of the visible decay products with the momenta in all axes being equal to 0 for a system of particles. According to [5] model trained on this feature set gives the best results. For example ROC AUC metric for discrimination between scalar and pseudoscalar hypothesis is 0.781.

3 Weights of events

3.1 Curve fitting to weights of events for different mixing angle hypothesis

Each event in dataset consists of kinematics data about output particles after collision and label which is weight of this event for given CP mixing angle (ϕ_{CP}) $(0\pi; 0.1\pi; 0.2\pi; 0.3\pi; 0.4\pi; 0.5\pi; 0.6\pi; 0.7\pi; 0.8\pi; 0.9\pi; 1\pi)$.

where ϕ_{CP} is mixing angle, equals 0 when Higgs boson is scalar, $\frac{\pi}{2}$ when it is pseudoscalar, and again π when it is scalar.

Let define mixing angle parameter ϕ as $\phi = 2\phi_{CP}$, so there is periodicity on this parameter in the range $(0, 2\pi)$.

Equation below describes weights of events for different mixing angle [3]:

$$w = A + B\cos(\phi) + C\sin(\phi) \tag{1}$$

For each event parameters A,B,C depends on the kinematics and correlations between tau lepton decay products.

Using available data, parameters A, B, C for each event where fitted using Levenberg-Marquardt algorithm as implemented in MINPACK (which is used by python scipy python package). Examples of fitting results are on figures 1-6. Blue dots corresponds to input data, and orange line is determined by equation (1) and fitted parameters A, B, C. Error is calculated using estimated covariance of fitted parameters returned by algorithm, it is defined as square root of elements on the diagonal of the covariance matrix.

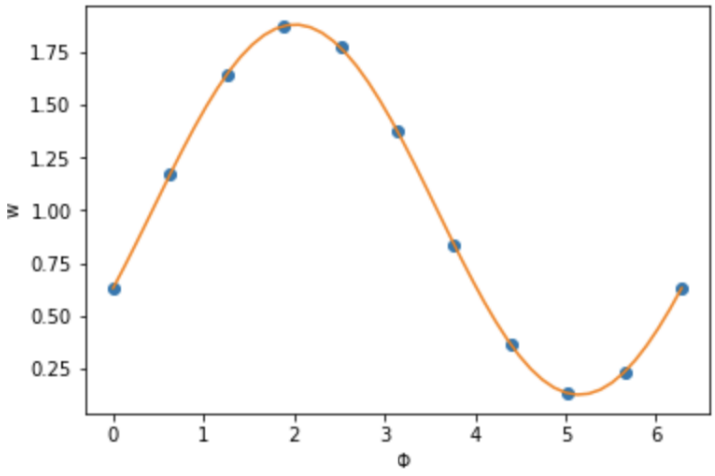

Figure 1: Fitted curve to weights of event for different mixing angle. One-sigma error of parameters A,B,C is $6.8 \cdot 10^{-8}$, $9.3 \cdot 10^{-8}$, $10.1 \cdot 10^{-8}$.

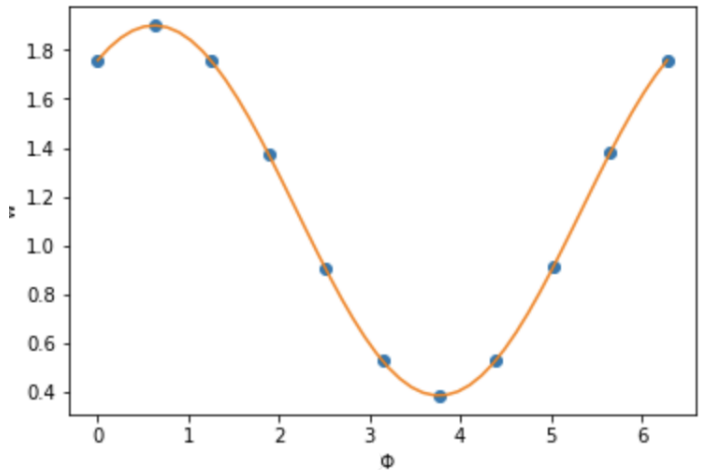

Figure 2: Fitted curve to weights of event for different mixing angle. One-sigma error of parameters A,B,C is $5.4 \cdot 10^{-8}$, $11.2 \cdot 10^{-8}$, $11.1 \cdot 10^{-8}$.

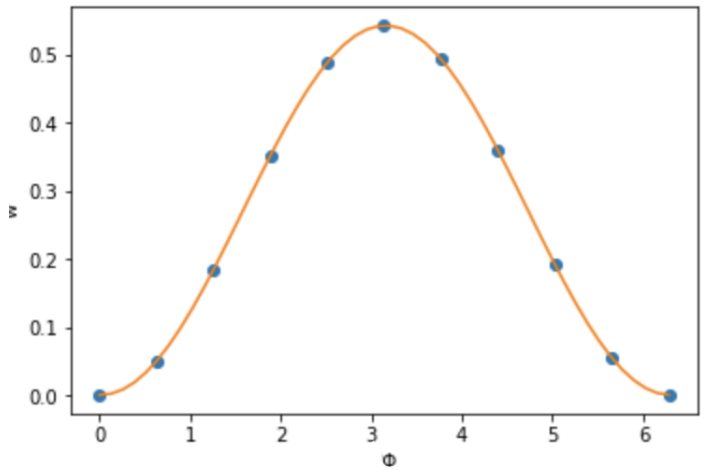

Figure 3: Fitted curve to weights of event for different mixing angle. One-sigma error of parameters A,B,C is $7.8 \cdot 10^{-8}$, $8.3 \cdot 10^{-8}$, $9.2 \cdot 10^{-8}$.

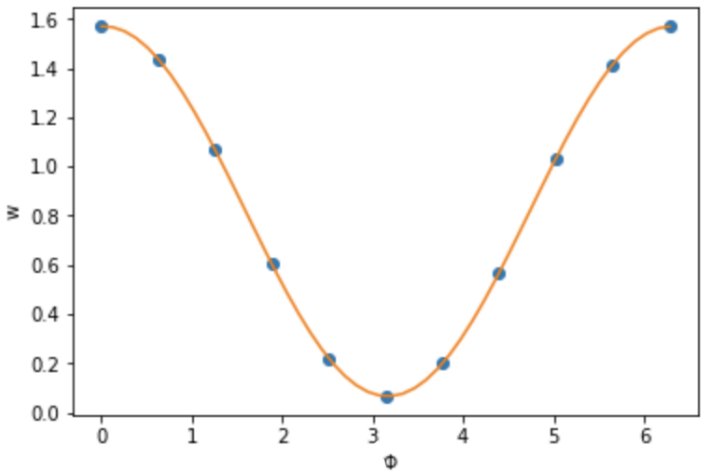

Figure 4: Fitted curve to weights of event for different mixing angle. One-sigma error of parameters A,B,C is $4.5 \cdot 10^{-8}$, $7.3 \cdot 10^{-8}$, $8.5 \cdot 10^{-8}$.

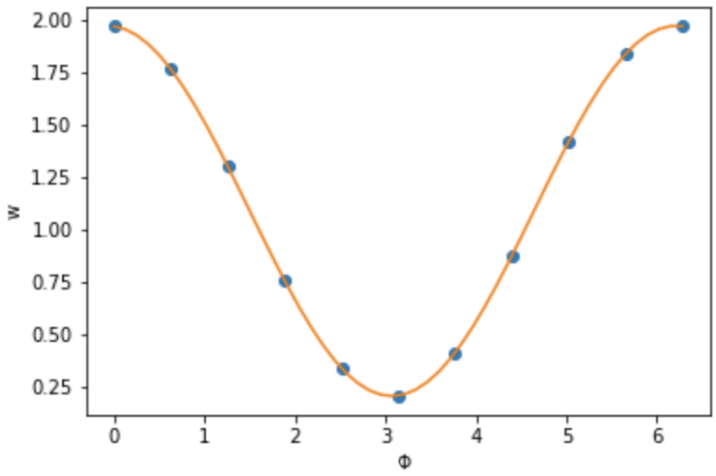

Figure 5: Fitted curve to weights of event for different mixing angle. One-sigma error of parameters A,B,C is $3.4 \cdot 10^{-8}$, $12.4 \cdot 10^{-8}$, $6.7 \cdot 10^{-8}$.

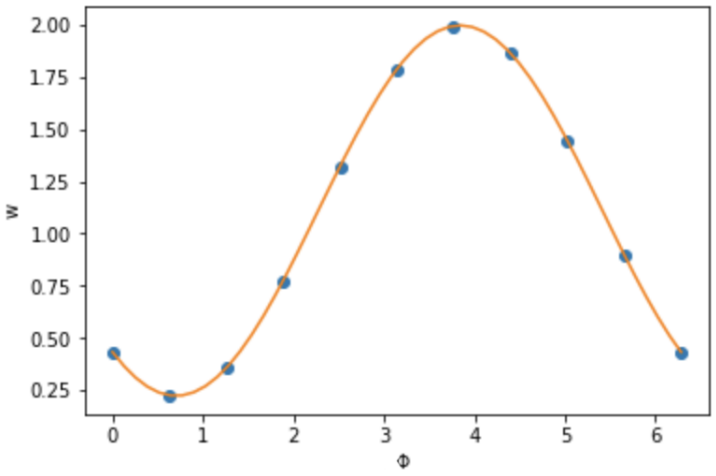

Figure 6: Fitted curve to weights of event for different mixing angle. One-sigma error of parameters A,B,C is $7.3 \cdot 10^{-8}$, $5.4 \cdot 10^{-8}$, $9.2 \cdot 10^{-8}$.

3.2 Unweighting events

Dataset of events can be reduced to only one mixing angle (chosen from mixing hypothesis) by unweighting process. It is performed by calculating weight w which corresponds to chosen mixing angle for each event. Given function 'unweight(w)'

```
def unweight(w):
    u = random.random() # sample from uniform (0,1)
    return 0 if w < u * 2 else 1
```

event is assigned weight 1 when function returns 1 and 0 when function returns 0.

For example given mixing angle equal 0.3π, histogram of acoplanar angle (with $y1 * y2 > 0$) of events that were assigned with weight 1 is shown in fig. 7

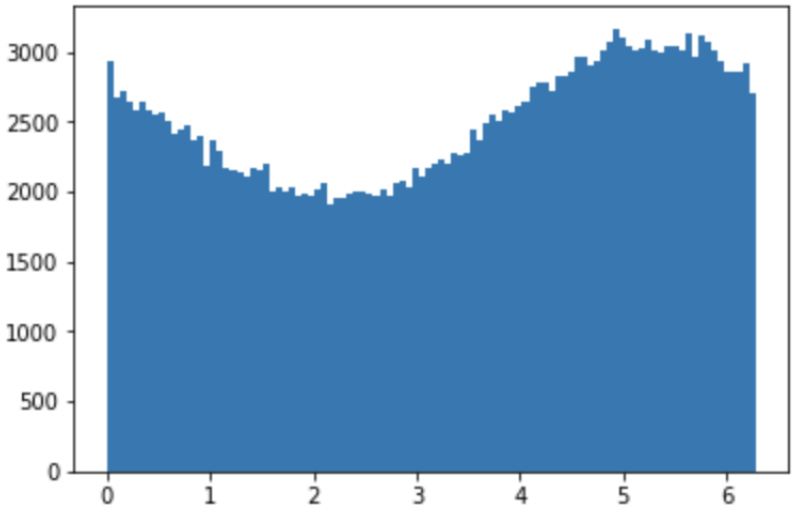

Figure 7: Histogram of acoplanar angle for unweighted data

Code can be found in notebook MixingAngleAnalysis.ipynb in section Acoplanar angle.

4 Multiclass classificaton

Multiclass classification is problem of classifying instances into one of three or more classes. In such type of problems the algorithm should predict a discrete label for a given example.

4.1 Model architecture

Usage of multiclass classificator based on neural network was chosen as one of technique to be tested. This choice is based on the work presented in paper [5]. It was proven that neural network trained as a binary classificator can successfully discriminize between Higgs boson scalar and pseudoscalar hypothesis. In this study we show, that such classificator can discriminate also between other mixed Higgs CP state hypothesis.

The neural network architecture is based on the architecture proposed in [5]. The network consisted of 6 hidden layers, 300 units each and was initialized with random weights with were optimized with stochastic gradient descent algorithm Adam. The model used RELU non-linearities after each of hidden layers. The only difference is in last layer. In the case of binary classification it was layer with one output with sigmoid activation to squish it to [0,1] interval, then interpreted as a probability of one of hypothesis. In the case of multiclass classification it

was exchanged for layer with N output (where N is number of hypothesis) with softmax activation.

Softmax activation [1] σ is described by equation (2) (z_i is i-th output of last neural network layer before activation).

$$\sigma(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{N} e^{z_j}} \tag{2}$$

One of features of this functions is that all outputs sum up to one. Therefore it can be interpreted as probabilities of belonging to one class.

One can observe that the softmax function is an extension of the sigmoid function to the multiclass case, as explained below. Let's look at the multiclass classification with N=2.

$$\sigma(z_1) = \frac{e^{z_1}}{e^{z_1} + e^{z_2}} = \frac{e^{z_1 - z_2}}{e^{z_1 - z_2} + 1} \tag{3}$$

Which is equation that describes sigmoid activation with $z_1 - z_2$ output of neural network. This method is therefore generalization of previous approach.

4.2 Valid distribution definition

In binary classification normalized weights (to sum up to one) of each event related to scalar and pseudoscalar hypothesis was chosen as valid distribution for classification (used directly in loss function as a valid distribution in cross entropy). In this generalized model as a valid distribution was chosen normalized weights related to each of hypothesis. Number N of CP mixed hypothesis related to equidistant mixing CP parameter ϕ. This weights were calculated using parameters fitted to weight function of each event as described in section 2.

It is described by code below:

```
def weight_fun(x, a, b, c):
    return a + b * np.cos(x) + c * np.sin(x)

classes = np.linspace(0, 1, num_classes) * 2 * np.pi

# Load calculated parameters A, B, C for each event
popts = np.load(os.path.join(data_path, 'popts.npy'))

for i in range(data_len):
    weights[i] = weight_fun(classes, *popts[i])
```

4.3 Loss function

Loss function used in stochastic gradient descent is described as a cross entropy of valid values and neural network predictions [1]. It is common choice in binary

and multiclass classification models (it was also used in binary classification described in [5]).

It is defined as follows:

$$L = \sum_{i=1}^{M} y_{i,c} \log(p_{i,c}) \tag{4}$$

where $p_{i,c}$ is value returned by i-th output of neural network for c-th event (so it is $\sigma(z_i)$ for c-th event), a $y_{i,c}$ is normalized value to probability of i-th weight of c-th event (sum of normalized weights for each event is 1).

This function is minimized when outputs of neural network corresponds to valid values.

Loss function in code is defined as follows:

```
sx = linear(x, "regression", num_classes)
self.preds = tf.nn.softmax(sx)

sum_ = tf.reshape(tf.reduce_sum(weights, axis=1), (-1, 1))

labels = weights / tf.tile(sum_, (1, num_classes))
self.loss = tf.nn.softmax_cross_entropy_with_logits(
                            logits=sx, labels=labels)
```

4.4 Results of experiments

In training used 800 000 rho-rho decay events. In testing used 100 000 events. The neural network was trained for 10 epochs.

In figure 8 is shown predicted values by neural network and valid distribution for multiclass classification with 11 classes. One can observe that despite in some cases neural network fails to predict valid difference between more and less probable class or weights distribution is moved by a few classes, it successfully predicts valid distribution.

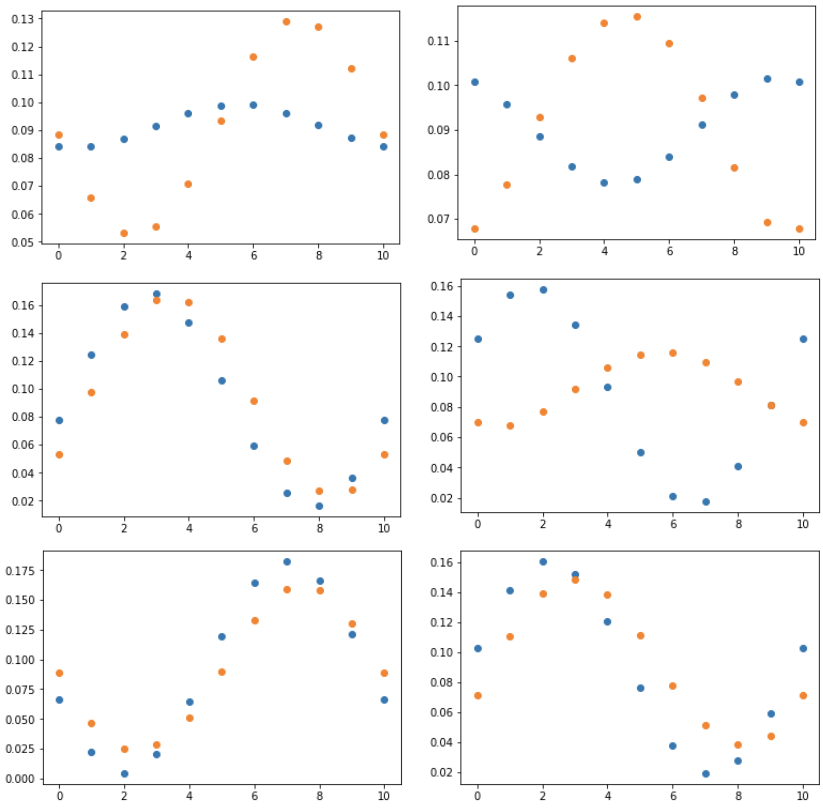

Figure 8: Comparison of values returned by neural network (orange) and valid distribution (blue). X axis corresponds to class number.

In order to quantically measure neural network prediction used accuracy metric. Assumed that event is correctly classified if difference between most probable predicted class and valid most probable class differs by no more than specific number of classes. Periodicity of classes is taken into account (for example distance between first and last class is zero). Let p_{max} be most probable predicted class, c_{max} be valid most probable class and N number of classes This distance d is defined as follows:

$$d = min(|p_{max} - c_{max}|, |(N-1) - p_{max} - c_{max}|, |-(N-1) - p_{max} - c_{max}|) \quad (5)$$

. To preserve direction between p_{max} and c_{max} (according to chosen minimum distance), sign of distance d correspond to sign of value $(p_{max} - c_{max})$ or $(N - p_{max} - c_{max})$, or $(-N - p_{max} - c_{max})$ (depending which one was chosen by minimum function in equation (5)).

In figure 9 there is shown distribution of difference between most probable predicted class and valid most probable class for different number of classes used to train classificator. It justifies such definition accuracy metrics.

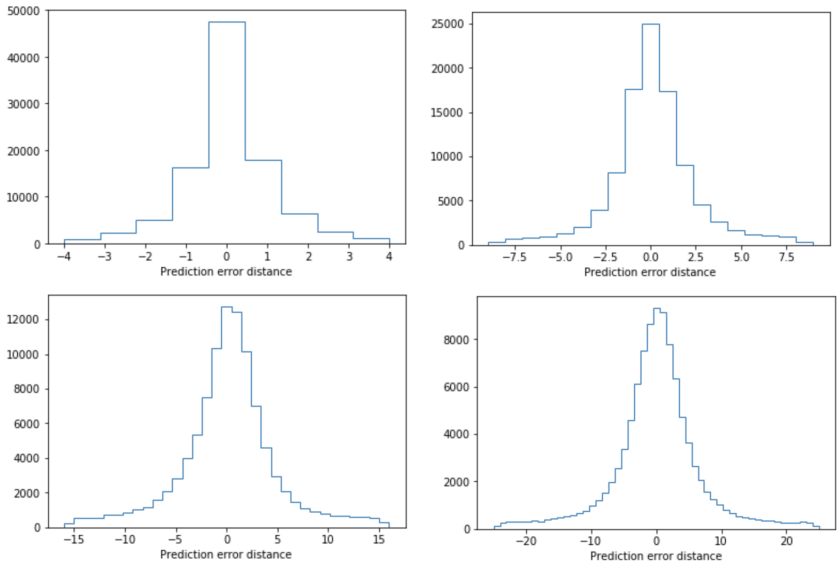

Figure 9: Distribution of difference between most probable predicted class and valid most probable class for different number of classes (8, 18, 32, 50 respectively).

Accuracy metric for different number of classes, and different maximum absolute error between most probable predicted class and valid most probable class. Let d_c be difference between most probable predicted class and valid most probable class (as defined in equation (5)) for c-th event, N be number of events and $\#(|d_c| < t_{max})$ number of events for which d_c is smaller than maximum distance d_{max}. Accuracy is defined as:

$$accuracy = \frac{\#(|d_c| < t_{max})}{N} \tag{6}$$

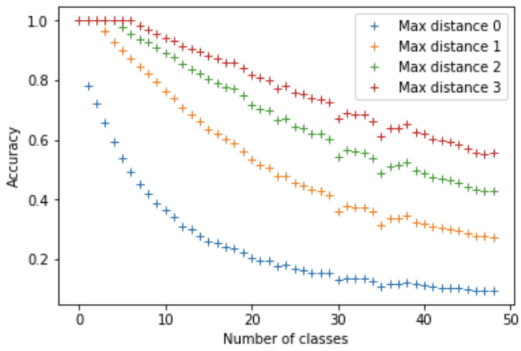

Figure 10: Accuracy metric for different number of classes, and different maximum absolute error between most probable predicted class and valid most probable class.

Accuracy is sensitive only to shifts of predicted most probable class. It justifies definition of another metric l1, which is mean absolute difference between predicted and valid distribution of weights for each class (equation 7)

$$l1 = \sum_{c=1}^{N} \sum_{i=1}^{M} |y_{i,c} - p_{i,c}| \, / (M \cdot N) \tag{7}$$

where $p_{i,c}$ is value returned by i-th output of neural network for c-th event, a $y_{i,c}$ is normalized weight of i-th weight of c-th event and M is number of classes and N number of events.

L1 metric results is presented in figure 11 for models trained with different number of classes.

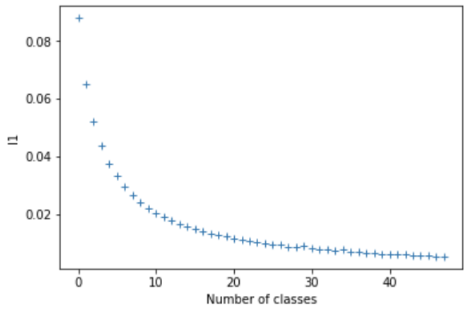

Figure 11: L1 metric for different number of classes

Similarly defined l2 metrics as defined in equation 8. Results are presented

in figure 12.

$$l2 = \sum_{c=1}^{N} \sqrt{\sum_{i=1}^{M} \left(y_{i,c} - p_{i,c}\right)^2 / M / N} \tag{8}$$

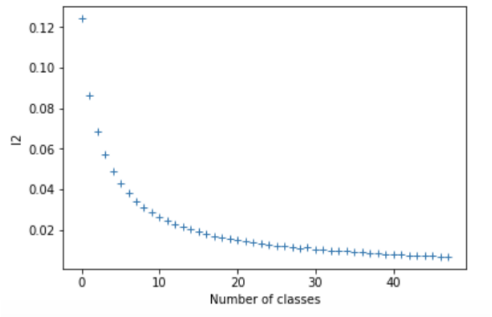

Figure 12: L2 metric for different number of classes

To compare with binary classification model ROC AUC scores where calculated to discriminate between two different classes. For testing there were chosen model with 21 classes, so CP mixing angle 0.1π corresponds to 2 class, 0.2π to 4 class etc.

CP angle	ROC AUC MULTICLASS	MAX ROC AUC	BASELINE ROC AUC
0.0π	0.500	0.500	0.500
0.1π	0.519	0.521	0.556
0.2π	0.571	0.577	0.592
0.3π	0.636	0.645	0.663
0.4π	0.688	0.701	0.690
0.5π	0.709	0.722	0.702
0.6π	0.690	0.701	0.690
0.7π	0.637	0.645	0.663
0.8π	0.573	0.577	0.592
0.9π	0.520	0.521	0.556
1.0π	0.500	0.500	0.500

Table 1: Comparison of values ROC AUC for discrimination between hypothesis CP mixing=0 and other hypothesis. ROC AUC MULTICLASS is calculated using outputs of multiclass model, BASELINE ROC AUC is calculated using outputs of model trained only on chosen pair of hypothesis and MAX ROC AUC is maximum ROC AUC (assuming that model returns valid label for each event).

All defined metrics are implemented in Analyze_different_class_num.ipynb notebook

5 Regression to parameters of weights distribution

In next step examined method of predicting directly parameters of distribution of weights of events for different mixing angle.

Neural network architecture was the same as described in previous sections. The only difference is in last layer which has 3 outputs which corresponds to calculated parameters of distribution (A, B, C as defined in equation 1). Activation of this layer is linear.

Loss functions is mean squared error between calculated and predicted parameters of weights distributions [1]. Loss function and last layer in code is defined as follows.

```
self.popts # parameters A, B, C for each event
sx = linear(x, "regr", 3)
self.loss = tf.losses.mean_squared_error(self.popts, sx)
```

Comparison of calculated an predicted graph of distributions of weights is presented in the figure 13:

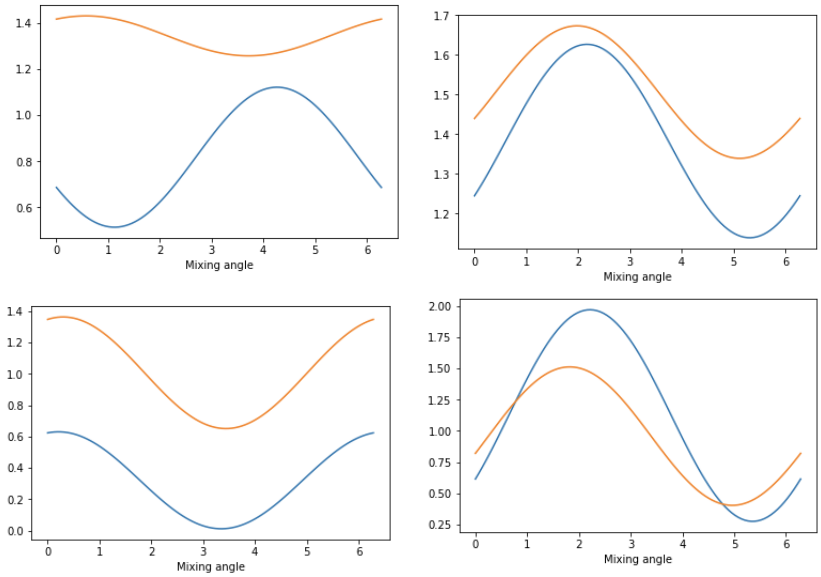

Figure 13: Comparison of graphs of calculated and predicted distribution of weights for different mixing angle. Blue is calculated distribution and orange is predicted.

These distributions were normalized (to make area under curve equal one)

and presented in figure 14. Integral of equation 1 is equal:

$$\int (a + b \cdot cos(x) + c \cdot sin(x))dx = ax + b \cdot sin(x) - c \cdot cos(x) \tag{9}$$

so definite integral from 0 to 2π is equal

$$\int_0^{2\pi} (a + b \cdot cos(x) + c \cdot sin(x))dx = 2\pi a \tag{10}$$

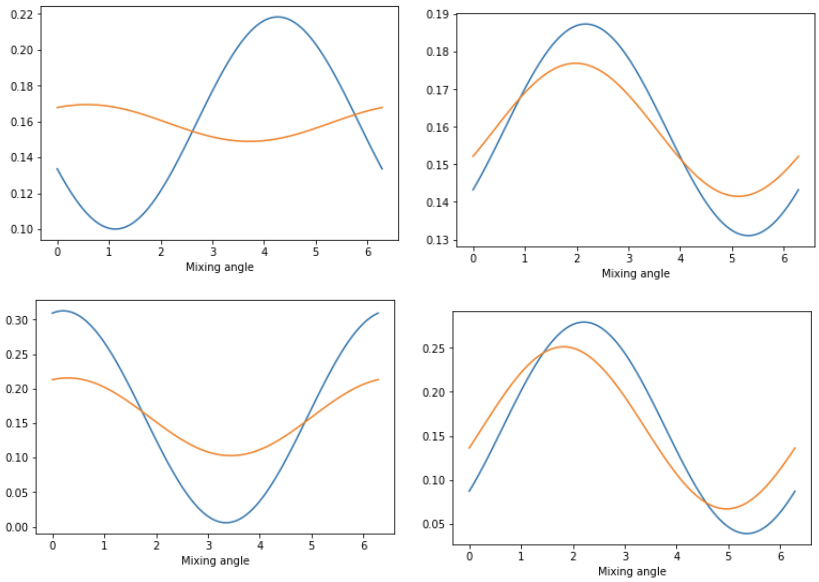

Figure 14: Comparison of graphs of normalized calculated and predicted distribution of weights for different mixing angle from figure 13. Blue is calculated distribution and orange is predicted.

In order to compare this method with multi classification approach, predicted and calculated parameters of distribution were used to calculate weights of discrete classes (for given number of classes).

Calculated classes were analyzed as in previous section. Firstly plotted histograms of distributions of difference between most probable predicted class and valid most probable class for different number of classes. Results are presented in figure 15

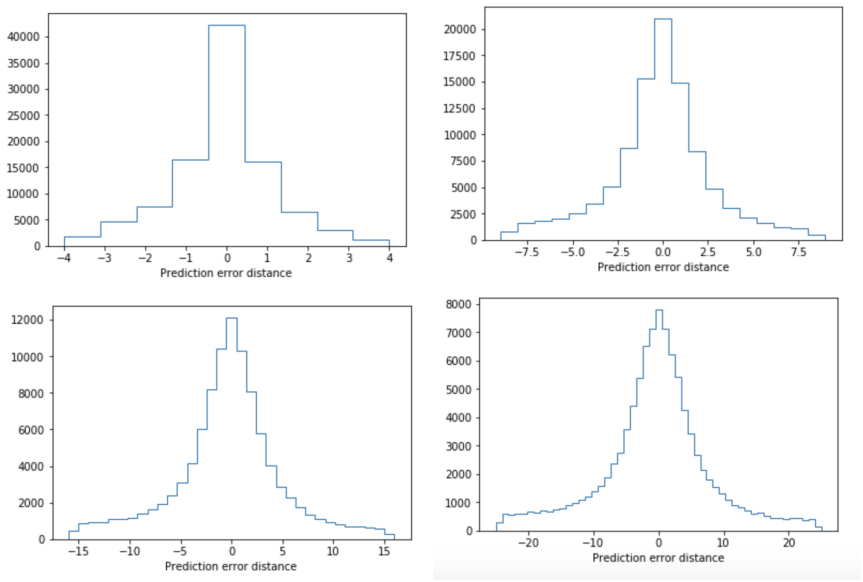

Figure 15: Distribution of difference between most probable predicted class and valid most probable class calculated based on predicted and calculated parameters of distribution of weights for different number of classes (8, 18, 32, 50 respectively).

Analogically calculated also accuracy metric for different number of classes, and different maximum absolute error between most probable predicted class and valid most probable class

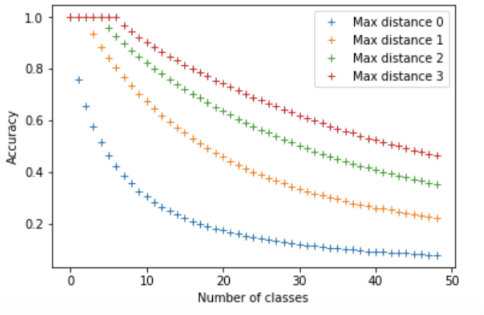

Figure 16: Accuracy metric for different number of classes, and different maximum absolute error between most probable predicted class and valid most probable class of weights of classes calculated using parameters of distribution

Analogically calculated also L1 and L2 metric:

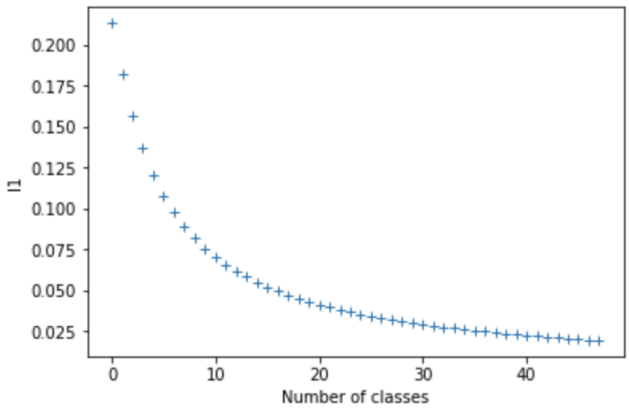

Figure 17: L1 metric for different number of classes for regression model

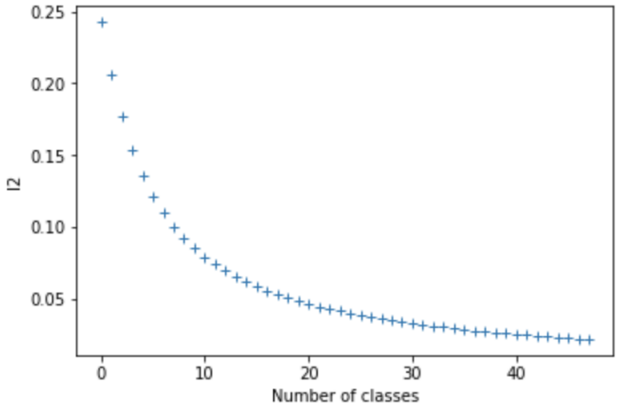

Figure 18: L2 metric for different number of classes for regression model

In this method it is also possible to calculate distance between most probable predicted mixing angle and most probable calculated mixing angle (as continues variable). Periodicity of mixing angle was taken into account (for example distance between 1.9π and 0.1π is 0.2π). Definition of this distance d is analogical to definition in previous section (in this case p_{max} and c_{max} are not most probable class number but most probable mixing CP parameter):

$$d = min(|p_{max} - c_{max}|, |2\pi - p_{max} - c_{max}|, | - 2\pi - p_{max} - c_{max}|) \qquad (11)$$

Histogram of this distance is presented in the figure 19.

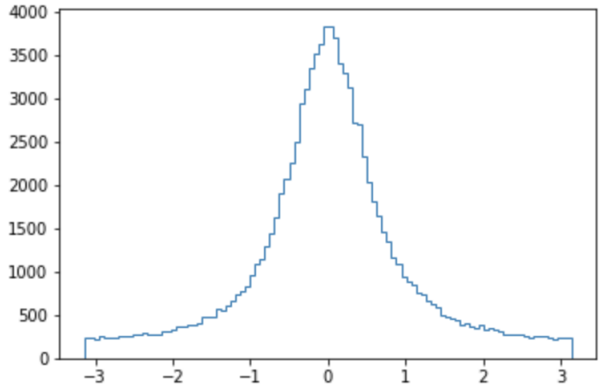

Figure 19: Distribution of distance between predicted most probable angle and valid most probable angle

6 Compare multiclass and regression approach

. Presenting accuracy metric defined in previous sections (and presented in figures 10 and 16) in one figure 20 one can observe that accuracy of multiclass and regression model is almost the same for max distance equal 0. If max distance is greater than 0 multiclass method gives slightly better accuracy than regression model. It shows that multiclass classification method returns predictions that are focused closer to true value.

It proves that neural network evaluated with accuracy metric return better results when weights of events are provided directly.

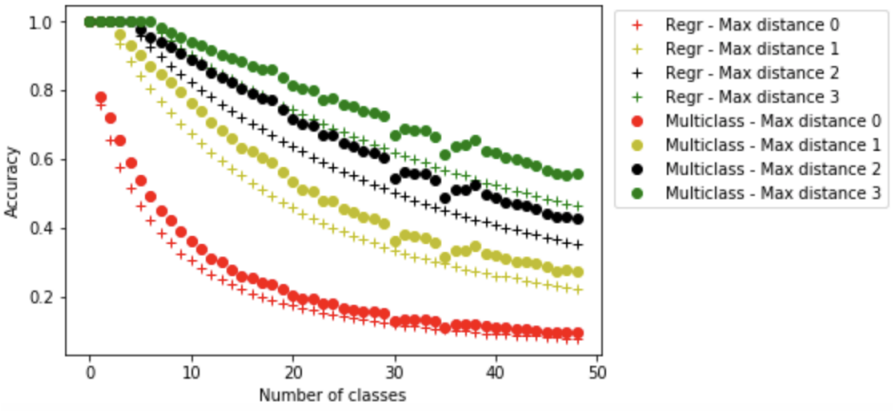

Figure 20: Accuracy metric for different number of classes, and different maximum absolute error between most probable predicted class and valid most probable class of weights for regression and multi-class method.

Comparing l1 metric (defined as in previous sections) it is difficult to spot the difference. It proves that differences between these two methods returns similar results not only for most probable mixing angle. L1 metric of these two models is presented in figure 21.

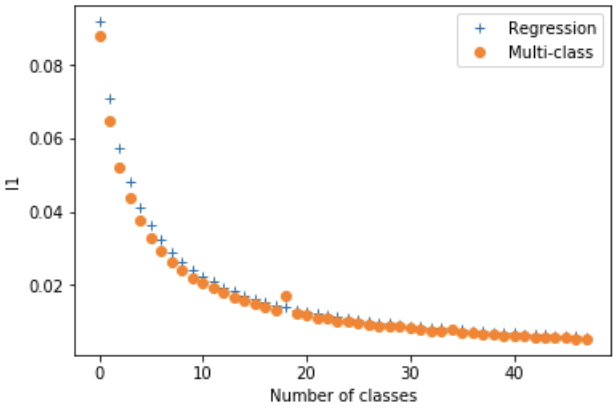

Figure 21: L1 metric for different number of classes for regression and multi-class model.

Comparison of this metrics shows that multi-class model works better when we reduce regression model to predict outputs for specific classes. However regression model doesn't require choosing the number of classes. Returned value allow to estimate whole distribution of weights for any hypothesis. In some cases it may be more desirable.

7 Conclusions

The study investigated application of neural networks in the measurement of CP state of Higgs boson in decay $H \to \tau\tau$ in $\rho\rho$ channel. The goal was to provide method of continues classification for different mixing CP angles.

Two different approaches were tested. Multiclass classification in which neural network predicts probabilities of event belonging to some CP mixing angle class. Regression to parameters of events' weights distribution in which neural network directly predicts parameters of CP mixing angle weights distribution for each event. There were also examined approaches of regression to most probable mixing angle which failed to solve this task correctly.

Multiclass classification approach proved to return better results than regression to mixing angle distribution parameters. However it requires defining the number of classes which corresponds to different mixing angle (which affects precision of results). In case when it is impossible to define certain certain number of classes it is better to use regression to mixing angle distribution parameters method.

Appendices

A Regression to most probable mixing angle

We examined also regression to most probable mixing angle. Unfortunately such defined neural networks fails to solve this task. In this section we present results of this experiment.

A.1 Most probable mixing angle

For each event exists most probable mixing angle value (which corresponds to maximum weight). This value is found based on parameters of fitted sine curve of mixing angle hypothesis weights. Derivative of eq. 1.

$$\frac{dw}{d\phi} = Ccos(2\phi) - Bsin(2\phi) \tag{12}$$

Derivative is equal to zero if

$$tan(2\phi) = \frac{C}{B} \tag{13}$$

so

$$2\phi = arctan(\frac{C}{B}) \tag{14}$$

Most probable mixing angle is found by comparing weights corresponding to 2ϕ, $2\phi + \pi$, $2\phi + 2\pi$ angles (if angle is in range $(0, 2\pi)$.

Given unweighted data to mixing angle 0.3π histogram of most probable mixing angle is shown in fig. 22

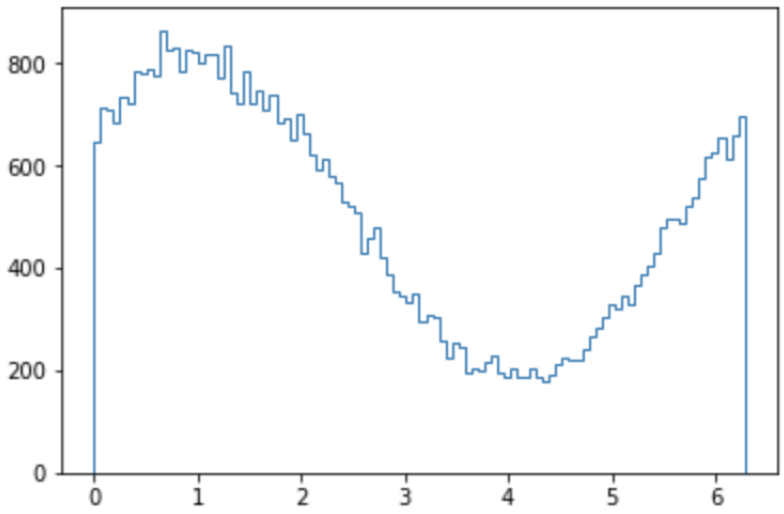

Figure 22: Histogram of most probable mixing angle of unweighted data

Code can be found in notebook MixingAngleAnalysis.ipynb in section Compare labels.

A.2 Sinus - cosinus parametrization

Training of neural network is performed to find directly most probable mixing angle parameter. As target value (mixing angle parameter ϕ) is periodic, this periodicity must be incorporated into the loss function. Otherwise distance between $phi = 0.1\pi$ and $phi = 1.9\pi$ would be computed to 1.8π while the true distance is 0.2π.

As proposed in [3] this is solved by redefining the target variable from a scalar to a two-component vector, where one component equals $sin(\phi)$ and second component equals $cos(\phi)$

Neural network architecture is the same as defined in section 4. The only difference is in last layer which has two outputs which corresponds to two elements of target vector. Activation of this layer may be linear, linear with clipping or hyperbolic tangent (tanh).

The input features of neural networks are defined in section 2.

A.3 Loss function definition

The loss function quantifies the distance between elements of true target vector and elements predicted by neural network. This distance is minimized during training.

As proposed in [3] we use Huber loss function [2], defined as:

$$L(x) = \begin{cases} \frac{1}{2}x^2 & if \quad |x| < d \\ d(|x| - \frac{1}{2}d) & otherwise \end{cases} \tag{15}$$

where d is small number and $x = y_i - p_i$ where y_i is true value and p_i is predicted value.

It is combination of mean squared error and mean absolute error which are typically used in regression tasks. It solves disadvantage of mean squared error which is heavily affected by outliers. On the other mean absolute error may be less efficient [6].

A.4 Training results

In training used 800 000 rho-rho decay events. In testing used 100 000 events. The neural network was trained for 10 epochs.

Neural network was testes using unweighted data. Three mixing angle parameter used in unweighting process (defined in sections 3.2) were chosen. In all histograms blue color corresponds to unweighted data to mixing angle parameter 0.1π, orange 0.5π and green 0.9π. In histograms are showed only events with assigned weight 1.

In figure 23 there is shown histogram of most probable mixing angle of unweighted data.

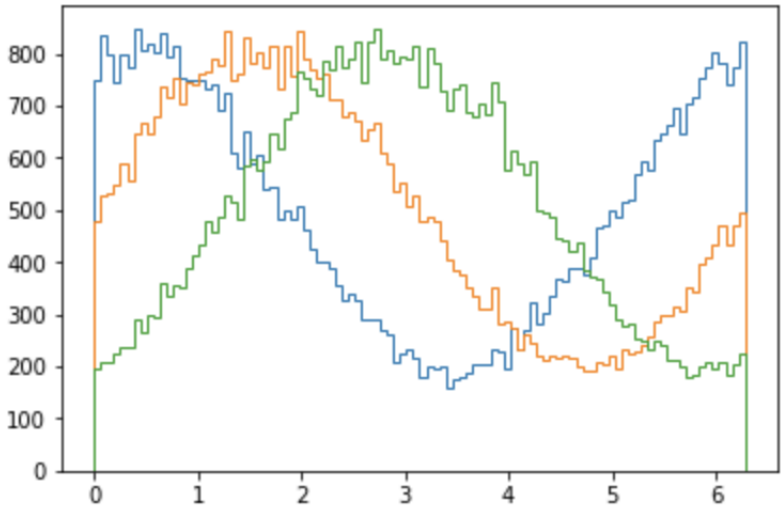

Figure 23: Histogram of most probable mixing angle of unweighted data

In figure A.4.1 and A.4.1 there is shown histogram of cosinus and sinus of this value respectively (which are defined as target vector).

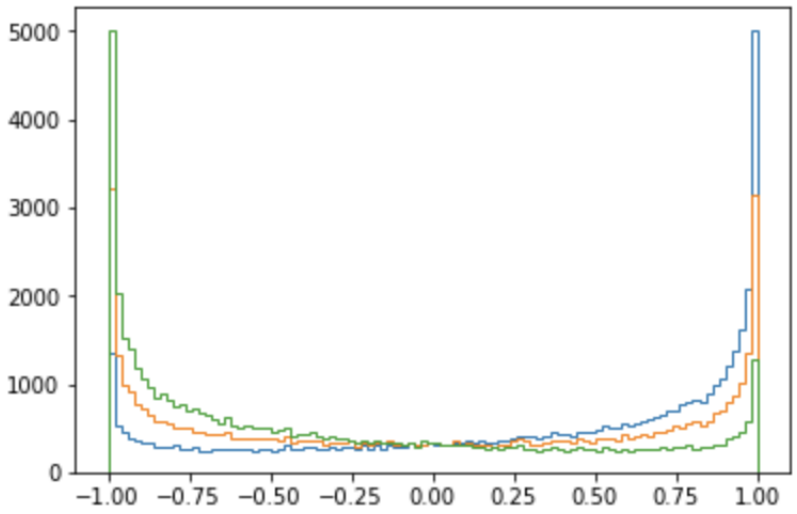

Figure 24: Histogram of cosinus of most probable mixing angle of unweighted data

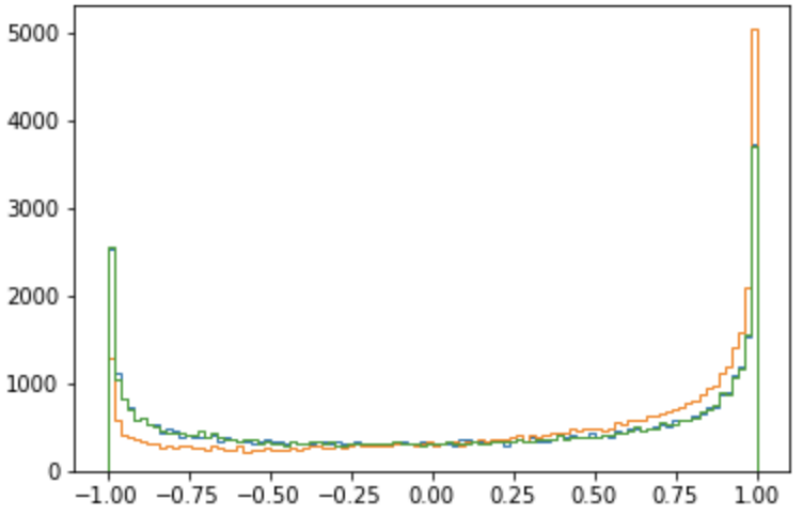

Figure 25: Histogram of sinus of most probable mixing angle of unweighted data

One can observe that most of the values are close to -1 and 1. This makes training of neural network difficult because it is much more important to predict precisely values close to -1 and 1 than in other regions. Three different activations in last layer were tested to solve this task.

A.4.1 Linear activation

In figures 26 and 27 are presented histograms of predicted values for unweighted events (as defined above). One can observe that neural network sometimes predict values smaller than -1 and greater than 1 which cannot be interpreted as sinus or cosinus of mixing angle parameter. Furthermore the distributions of predicted values are not correct (in comparison to distributions presented in figures and .

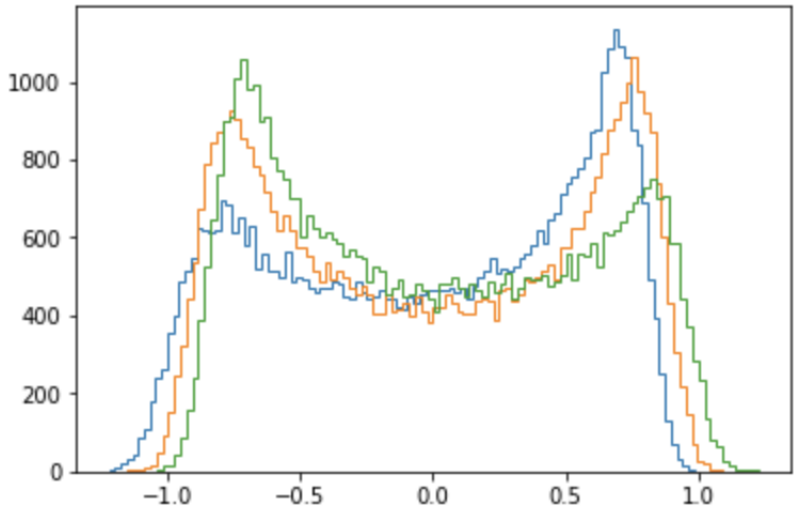

Figure 26: Histogram of predicted cosinus value for linear activation

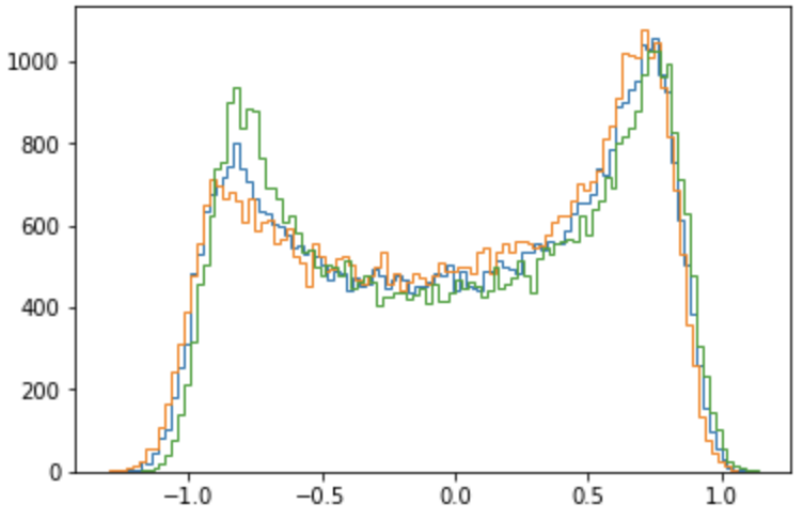

Figure 27: Histogram of predicted sinus value for linear activation

A.4.2 Linear activation with clipping

In order to keep values in range [-1, 1] clipping is added to activation in last layer. Values greater than 1 are clipped to 1, and smaller than -1 to -1. Results are presented on histograms below. One can observe that this modification of activation doesn't solve problem with incorrectly predicted distribution of values.

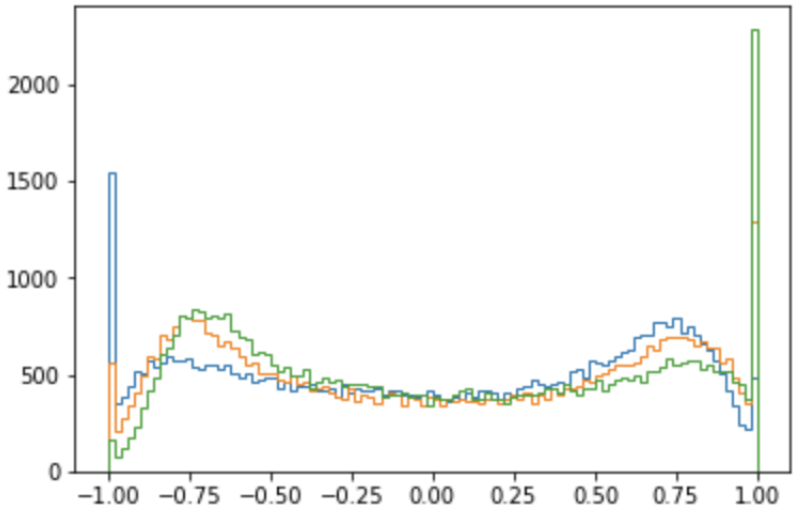

Figure 28: Histogram of predicted cosinus value for linear activation with clipping

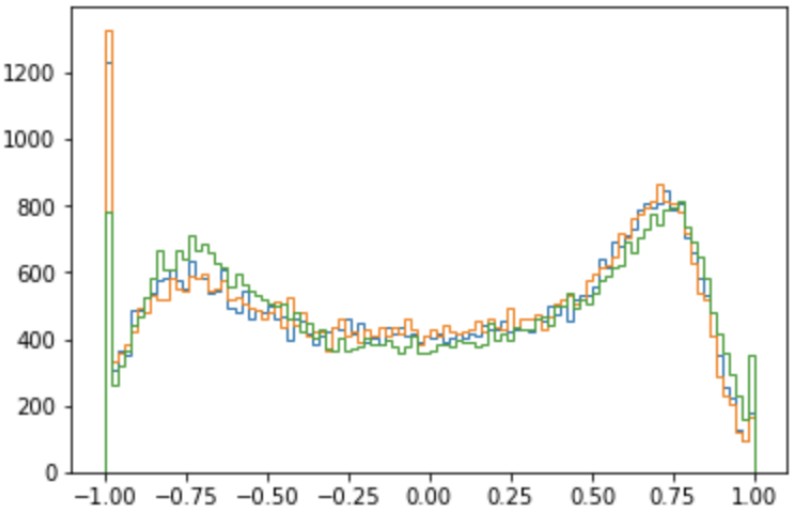

Figure 29: Histogram of predicted sinus value for linear activation with clipping

A.4.3 Tanh activation

Hyperbolic tangent were also tested as activation function since its image is [-1, 1]. Histograms of predicted values are presented in figures below. One can observe that it solved the problem of values greater or smaller than one. Unfortunately neural network still too often predicts value close to 0.

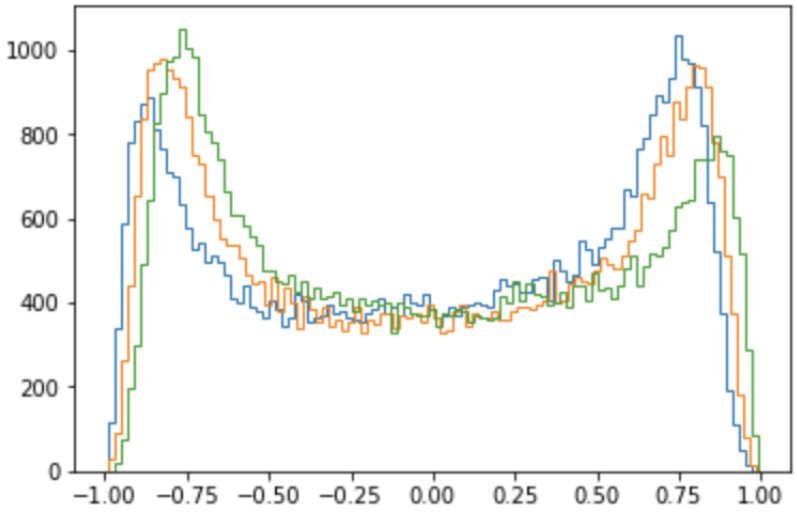

Figure 30: Histogram of predicted cosinus value for linear activation with clipping

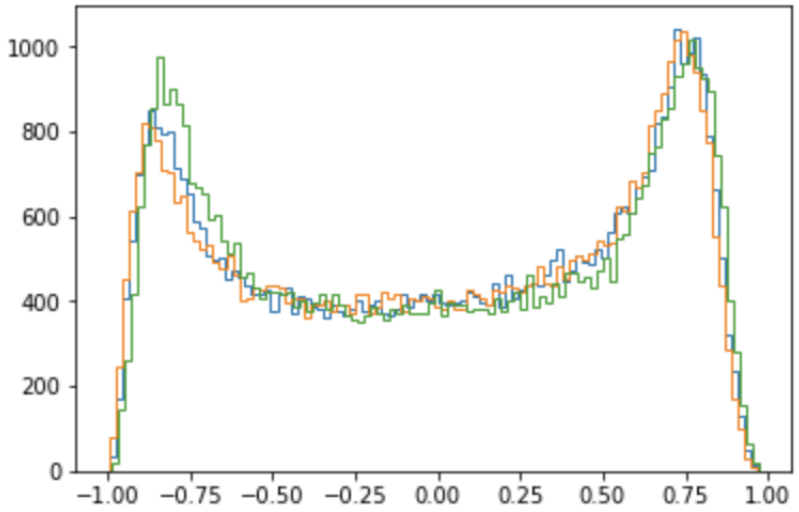

Figure 31: Histogram of predicted sinus value for linear activation with clipping

A.5 Parametrized regression to most probable mixing angle parameter summary

Neural network fails to predict values correctly with sinus cosinus parametrization. It is caused by specific distributions of target values. Neural network tends to incorrectly predict extreme values, which are majority in this case. It seems that tanh activation, but better parametrization is necessary.

References

[1] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[2] P. Huber. Robust estimation of a location parameter. *Ann. Math. Statist. Vol. 35, Num. 1 (1964)*, pages 73–101.

[3] S. Mæland. Pixel detector performance and study of cp invariance in h to tau tau decays with the atlas detector. *bora.uib.no:1956:18106*, pages 87–93.

[4] D. Perkins. *Introduction to high energy physics*. Press Syndicate of the Univerisity of Cambridge, Cambridge, United Kingdom, 2000.

[5] Z. Was R. Józefowicz, E. Richter-Was. Potential for optimizing higgs boson cp measurement in h to tau tau decay at lhc and ml techniques. *Phys. Rev. D 94, 093001 (2016)*.

[6] M. Sugiyama. *Introduction to Statistical Machine Learning.* Morgan Kaufmann Publishers Inc., San Francisco, CA, 2016.