Improving Signal Classification for HNL with τ using Transfer ML

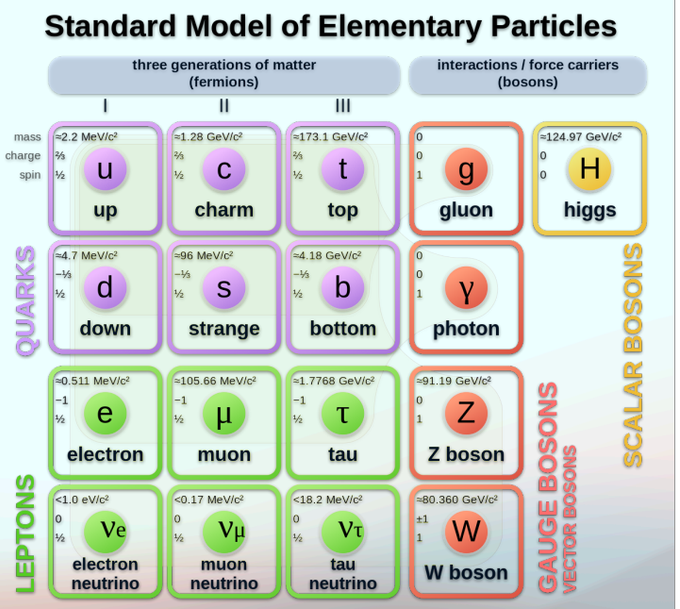
September 4, 2023

Dmitri Demler

Supervisor: Konstantin Androsov

Standard Model

- Successful at predicting interaction between particles
- Struggles to explain dark matter to the mass of neutrinos

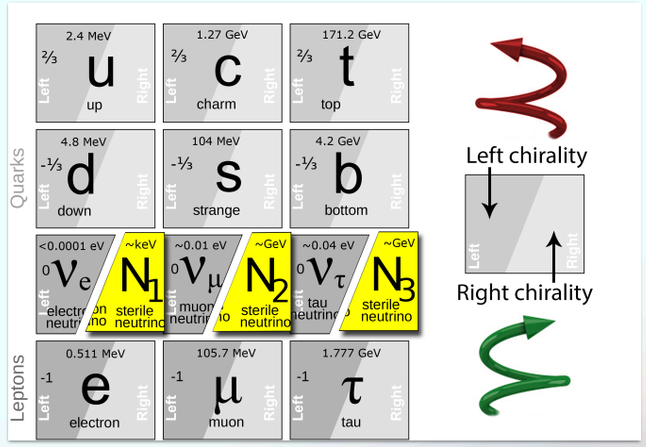


The Standard Model theory has been remarkably successful in predicting the interactions between particles. However, as with any scientific theory, the Standard Model has its limitations of not being able to describe absolute phenomena such as neutrino mass and dark matter.

- *Transition:*
    - *To address these limitations, let's look at a proposed extension: the neutrino Minimal Standard Model.*

νMSM

- Neutrino minimal standard model
- Heavy Neutral Leptons
  - Introduces three right-handed, colorless neutrinos
  - Don’t interact electromagnetically or via strong force



One proposal is the neutrino Minimal Standard Model which can explain these phenomenas.
This theory introduces three right-handed neutrinos known as Heavy Neutral Leptons.
Neutrinos in the standard model are only left handed so that they can be massless. vMSM, however, introduces right handed colorless neutrinos that won’t interact electromagnetically or via the strong force. Due to this, it will not be a part of any standard model interaction. The only way to notice it is through standard model neutrinos. We focus on tau neutrino interactions.

In the standard model there are only left handed neutrinos while vMSM introduces right handed colorless neutrinos that won’t interact electromagnetically or via the strong force. Due to this, it will not be a part of any standard model interaction. The only way to notice it is through standard model neutrinos. We focus on tau neutrino interactions.

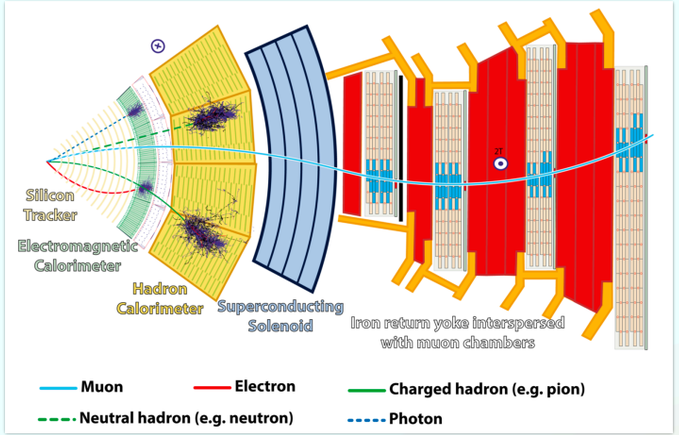
- *Understanding HNLs requires specialized techniques. Let's explore the methods and data we use.*

~~However, their presence can help explain why neutrinos have mass. Through a mixing process (between light and heavy neutrino states) often referred to as ‘*seesaw mechansim*’, HNL’s can account for the observed neutrino masses.~~

~~This was originally done to explain why we only observe neutrinos in a left-handed helicity state. This observation was crucial for developing the V-A structure of the weak interaction.~~
~~Since neutrinos are considered massless in the model, their chiral state—which is a property that doesn't change even if you change your point of view—coincides with their helicity state. This means that right-handed neutrinos would not interact through the weak force.~~
~~However this view was challenged when neutrino oscillations were observed and indicated that at least two types of neutrinos have non-zero mass.~~

Our task

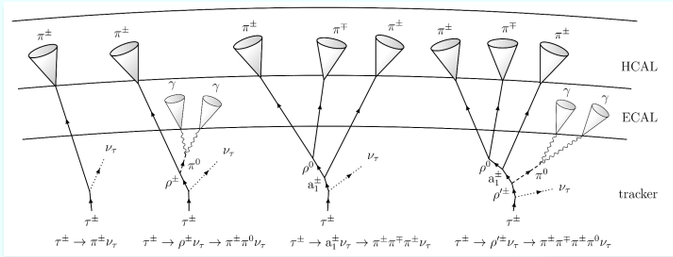
- Study prompt decays of HNL
  - Kinematic signatures
- Distinguish HNLs and background SM processes with similar signatures in the detector.
- Use processed 2018 CMS data

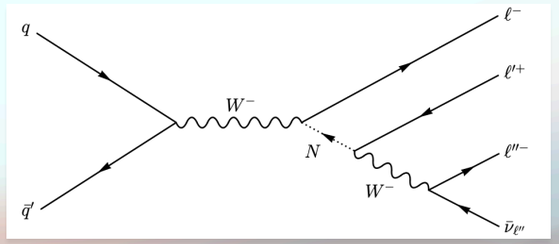


In our work, we study the prompt decays of these HNLs and the unique kinematic signatures they produce. By using these kinematic values we hope to discern HNL decays from other standard model processes.
To accomplish this we used CMS 2018 data which was processed by Paul. The compact muon Solenoid detector (CMS) aims to explore physics at the TeV scale and search for phenomena Beyond the Standard Model.
Analyzing these signatures requires sophisticated methods, and this is where machine learning techniques can prove to be useful.

Focus on τ Leptons and HNLs

- Focus
  - HNLs to 3 lepton decay
  - Targeting |VτN|
- Importance of Hadronic τ's
  - 65% decay into hadrons
- CMS sensors
  - Tracker for charged particles
  - ECAL for e & γ
  - Hadronic calorimeter for hadrons





Dmitri Demler

This work focuses on Heavy Neutral Leptons (HNLs) to 3 leptons decay at CMS, focusing on events with at least one hadronically decayed τ among three final-state leptons.

Why are hydronic τ's crucial? Because the large mass of the τ lepton compared to other standard model leptons allows the τ to decay into hadrons. Specifically, τ leptons decay into hadrons and a neutrino about 65% of the time.

To identify these particles we use CMS's various sensors. For electrons, we look at the tracks in the inner tracker and the energy deposited in the Electromagnetic Calorimeter, or ECAL. Photons, being neutral, don't leave a track but still deposit energy in the ECAL.

Hadrons on the other hand like pions and kaons oftentimes go past the ECAL and end up being detected in the hadronic calorimeter.

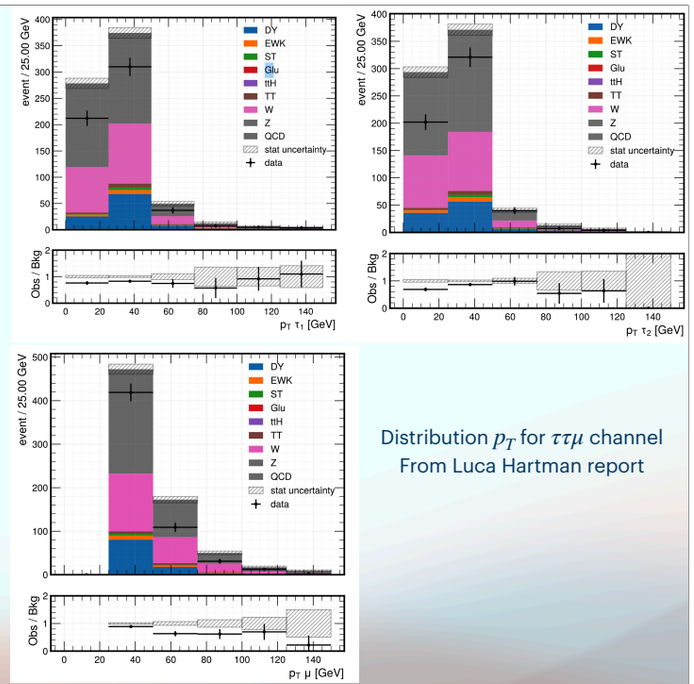
By combining the kinematic values of these different sensors we are able to get the 4-momentum of each particle and thus use them to differentiate HNLs

~~Since the final state can have different particles in the final state, various sensors are used to obtain the data.~~
~~Electrons are identified through the tracks they leave in the inner tracker and the energy they deposit in the Electromagnetic Calorimeter (ECAL). Photons, being neutral, don't leave a track in the tracker but deposit most of their energy in the ECAL. The energy of electrons and photons is reconstructed similarly, as electrons often emit photons (bremsstrahlung), which then typically produce electron-positron pairs. Both electrons and photons may reach the ECAL as small clusters, and all particles must be considered for energy computation.~~

~~Muons leave a distinct signature in the detector, usually passing through all calorimeters to reach the muon chambers. If an outer track in the detector matches a track in the inner tracker, the muon's kinematics can be precisely reconstructed.~~

Event Preselection & Channels

- Event reconstruction
  - 3 well reconstructed isolated leptons
  - $\Delta R \leq 0.5$
- Channels and triggers
  - 5 channels: $ll'l'' \in \{\tau\tau\mu, \tau\tau e, \tau\mu\mu, \tau\mu e, \tau ee\}$
- Data preprocessing
  - Deep Tau discriminator score
- Lepton requirements
  - $p_T^{e,\mu} > 10$ GeV
  - $p_T^{\tau} > 20$ GeV



Distribution p_T for $\tau\tau\mu$ channel
From Luca Hartman report

- **Event Reconstruction**
  - We focus three well reconstructed and isolated leptons within the acceptance. Only particles within a cone of radius delta R =0.5 around jet axis are considered.
  - The CMS sensors, discussed in the previous slide, enable efficient reconstruction of jets and hadronic τ decays, precise identification of electrons and muons, and determination of missing transverse momentum.
- **Channels and Triggers**
  - Our study utilizes 5 different final state channels which are ττμ,ττe,τμμ,τμe,τee.
  - The trigger selection varies with decay channel, using single electron, single muon, or di-tau tiggers as appropriate.
- **Data Preprocessing**
  - We also applied preselection to retain physically plausible events by using a cut based on the Deep tau discriminator score and additonal cuts on the electrons and muons within jets.
- **Lepton requirements**
  - Three well-reconstructed and isolated leptons are required with energy thresholds of 10 GeV for electrons and muons and 20 GeV for τ leptons.

The image here shows the transverse momentum distribution for the 3 particle in the ττμ channel as an example.

Include about pt
3 well reconstructed and isolated leptons > 10 (electrons, muon), 20(for tau) GeV threshold
Events with jet B-tagged jets
Introduce the 5 channels

Using single electron, single muon, or di-tau trigger depending on channel

To reconstruct an event we choose b-tagged jets with long-lived particles that give the most information which are electrons, photons, charged pions, and muons. The CMS sensors I explained in the last slide allow for efficient reconstruction of jets and hadronic τ decays, determination of missing transverse momentum, and the identification of electrons and muons with high precision. We use 5 final state channels in the study shown here. Different triggers such as single electron, single muon, or di-tau triggers are used depending on the decay channel.

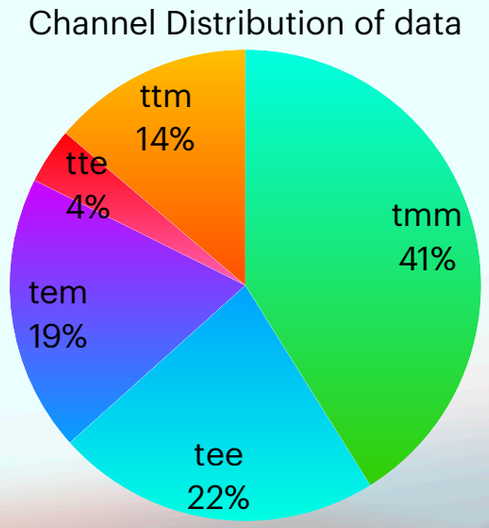
Preselection was performed to keep only physically possible events. A cut was performed using the DeepTau discriminator score against jets >=5. Furthermore, to remove electrons and muons produced within jets a cut is applied on their relative isolation score <0.15.

3 well reconstructed and isolated leptons are required to have GeV thresholds of 10 for electrons and muons, and 20 for tau.

The image here shows the distribution of the transverse momentum of (a) the first tau lepton, (b) the second tau lepton, and (c) the muon in the ττμ channel.

Dataset

- HNL Mass Hypothesis
  - $m_{\text{HNL}}^{\text{hyp}} \in [85, 1000]$ GeV
- Data Generation
  - MadGraph, Pythia, Geant4
- Data Size and Cuts
  - 216k signal events, 1.4M background
- Potential Challenges
  - Risk of overfitting
- Variable and Feature
  - m, p_T, ϕ, η
  - Weight, channel, $m_{\text{HNL}}^{\text{hyp}}$



- HNL mass hypothesis
  - Range of 16 HNL mass hypotheses, from 85 to 1000 GeV.
  - For signal data their value is what was specified in the simulation software, for background these values were randomly chosen.
- Data generation
  - Data produced using MadGraph for event generation, Pythia for tau decay, and Geant4 for particle interactions.
- Data Size and cuts
  - initial dataset of 1.6 million events reduced to about 216k signal events and approximately 1.4 million background events after applying various cuts.
- Potential Challenges
  - In the context of machine learning, the small number of signal events can cause problems due to the high overfitting chance
- Variables and Features
  - Kinematic variables include mass, transverse momentum, azimuthal angle, and pseudorapidity.
  - Additional event-specific variables include particle charge, event weight, channel, and HNL mass set in the Monte Carlo simulation (for signal events).

Now with this dataset in hand, lets look at previous attempts at classifying HNLs

This brings us to the dataset itself. We use a range of 16 HNL mass hypothesis starting from 85 to 1000 GeV. This data was produced using MadGraph and Pythia for tau decay and particle interactions with Geant4. The various cuts I described before reduces the signal data from 1.6 million events to about 216 k with about 1.4 million background events. In the context of machine learning, this many signal events is quite small and overfitting can prove to be a big challenge.

The variables used to describe the kinematics of the particles are its mass, transverse momentum, azimuthal angle, and pseudorapidity. Furthermore each event contains the charge of each particle, the event weight, channel, and HNL mass set in the Monte-Carlo simulation (for signal).

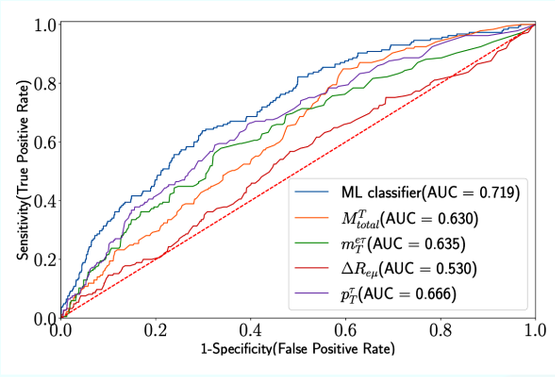
~~Dataset: num signal and background~~
~~Contains: kinematic features…~~
~~Still have a lot of background, how to do this?~~

~~Lucas Mallie tried using BDT~~

Previous Work

Lucas Mollier

- Machine Learning algorithm
  - XGBoost
- Training Approach
  - Classifiers for each mass and channel
- Inputs:
  - 40 classical observables



ROC curves and AUC values for HNL mass 250 GeV

Classical observables: calculated + raw kinematic variables (m, ΔR, M_T^{tot}, etc...)

Lucas Mollier started applying machine learning classifiers to see if they could improve classification methods. He used the XGBoost machine learning algorithm which uses binary decision trees to try to classify the data. He trained classifiers for each mass and each channel separately with the inputs of each model 40 classical observables. These observables consist of various physically important kinematic values. He then compared these XGBoost models with the four best classical observables.

As seen in these plots, the XGBoost model was able to perform better than the classical features but can we do better?

TODO: talk about his results

- Introduction:
  - "Lucas Mollier was to first to try and apply machine learning for data classification."
- Machine Learning Algorithm:
  - Utilized XGBoost, a machine learning algorithm based on binary decision trees, for data classification.
- Training Approach:
  - Trained classifiers for each mass and channel separately, using 40 classical observables as input features.
- Classical Observables:
  - These observables encompass various kinematically important variables that are physically significant.
- Comparison:
  - Benchmarked the performance of the XGBoost models against the four best classical observables.

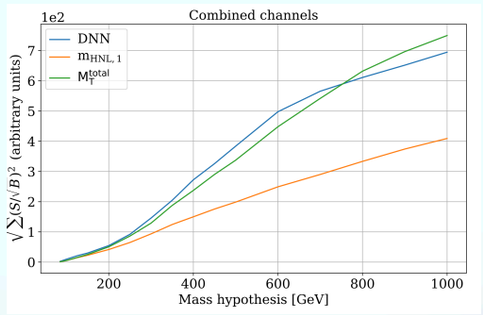
"*Having seen how machine learning was previously applied, let's explore how we can build upon this work.*"

~~such as the raw inputs, Angular distance (deltaR) and Tranverse mass(MT_total)~~

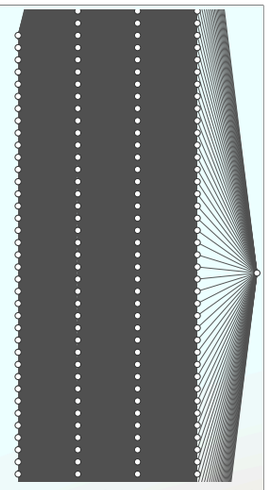
Previous Work

Nelson Glardon

- Machine Learning algorithm
  - Deep Neural Network
- Input:
  - 85 input features
- Training Approach
  - One classifier for all channels and m_{hyp}



Significance Estimator for DNN score



Visualization of the DNN model

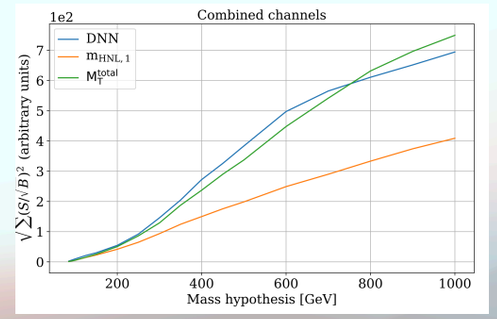
Nelson Glardon continued work on HNL classification by using a deep neural network instead. He used a set of 85 features mainly created out of the kinematic variables of the 3 leptons and the MET, similar to Lucas. He then tried to reduce the number of input features by removing the lesser important ones and tried different depths. Ultimately he found that the best model was using only the best 29 input features and 3 hidden layers.

Previous Work

Nelson Glardon

- Machine Learning algorithm
  - Deep Neural Network
- Input:
  - 85 input features
- Training Approach
  - One classifier for all channels and m_{hyp}

- Best Model Specifications:
  - Input: 29 features
  - Depth: 3
  - Width: 58
  - Optimizer: Adam
  - Dropout: 0.2



Significance Estimator for DNN score

Nelson Glardon continued work on HNL classification by using a deep neural network using tensorflow instead. He used a set of 85 features mainly created out of the kinematic variables of the 3 leptons and the MET, similar to Lucas. He then tried to reduce the number of input features by removing the lesser important ones and tried different depths.

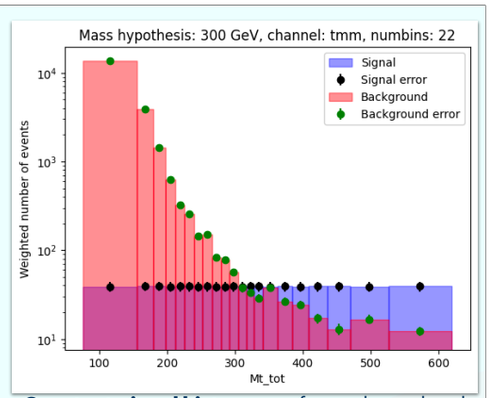
Ultimately he found that the best model was using only the best 29 input features, 3 hidden layers, and a width of 58. This model was able to perform slightly better than Mt_tot at low mass hypothesis but was worse at larger values where Mt_tot became more important. You can see this in this plot which shows significance (kinda) of different methods at the different mass hypothesis. This was a surprising result since in theory DNN should perform better than any one of its input features. If hypothetically Mt_tot is the only important feature, the model should learn to merely retain this feature throughout its layers.

This is where I come in.

Model comparison

Histograms

- ✳ Objective
  - Have a clear metric to compare various models and features at different mass hypotheses
- ✳ Histograms
  - Model scores (or features) vs Event count at a specific m_{hyp}
- ✳ Statistical Certainty
  - Relative weighted uncertainty for each bin
  - $\sqrt{\frac{\Sigma w^2}{\Sigma w}} < 0.15$



Constant-signal histogram of $\tau\mu\mu$ channel and m_{hyp} 300 GeV

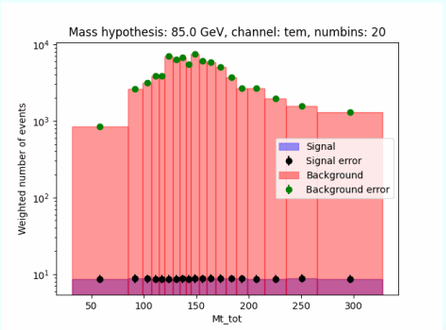
The first thing we did was to improve the method of comparing models. We need to see each models or features accuracy at classifying if an event is an HNL decay at different mass hypothesis. This is critical so that specific mass hypothesis are not prioritized accidentally. We used only signal data that corresponded with the mass hypothesis but used all background data.

To do this we first plotted histograms of signal and background at different x values (which are the model scores of feature values). As with most histograms, the y axis is the weighted sum of the number of events. Because each histogram is quite specific, since you plot one for each channel and each mass hypothesis value, limiting statistical uncertainty was key. We ensured that each bin had a relative statistical uncertainty less than 0.15.

Model comparison

Histogram Binning

- Objective
  - Have a clear metric to compare various models and features
- Histograms
  - Model scores (or features) vs Event count
- Statistical Certainty
  - Relative weighted uncertainty for each bin
  - $\sqrt{\frac{\Sigma w^2}{\Sigma w}} < 0.15$



Constant-signal histogram of $\tau e \mu$ channel of M_t^{tot} for different m_{hyp}

- Constant-signal histogram
  - Left to right
  - Signal heigh stays the same
  - Easy to compare at wide range of x values

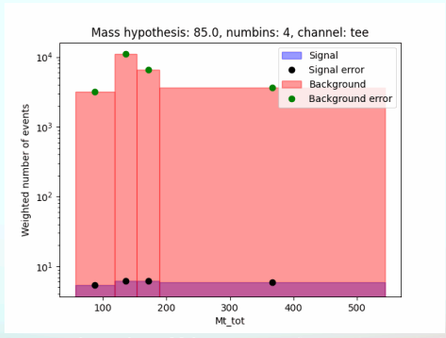
We implemented two different binning techniques that split the binning of the histograms differently. For the first method, we keep each bins signal height the same. Using this, we can start with a very large bin number and decrease it until statistical certainties are satisfied. This way one can simply focus on the background histogram to compare.

The second binning technique we implemented was by making bins from right to left. We widen the bin width until it reaches the statistical uncertainty threshold and then continue with the next one with each next bin having more weighted signal events than the previous one. This technique is more useful for ML models since it allows us to compare better the events that the models are very confident in.

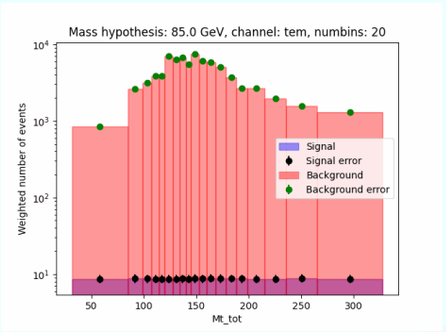
Model comparison

Histogram Binning

- increasing-signal histogram
  - Right to left
  - Signal height increases
  - Better comparison at high x-values



Increasing-signal histogram of $\tau e\mu$ channel of M_t^{tot} for different m_{hyp}



Constant-signal histogram of $\tau e\mu$ channel of M_t^{tot} for different m_{hyp}

- Constant-signal histogram
  - Left to right
  - Signal heigh stays the same
  - Easy to compare at wide range of x values

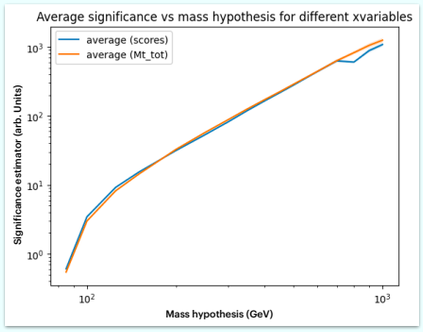
We implemented two different binning techniques that split the binning of the histograms differently. For the first method, we keep each bins signal height the same. Using this, we can start with a very large bin number and decrease it until statistical certainties are satisfied. This way one can simply focus on the background histogram to compare.

The second binning technique we implemented was by making bins from right to left. We widen the bin width until it reaches the statistical uncertainty threshold and then continue with the next one with each next bin having more weighted signal events than the previous one. This technique is more useful for ML models since it allows us to compare better the events that the models are very confident in.

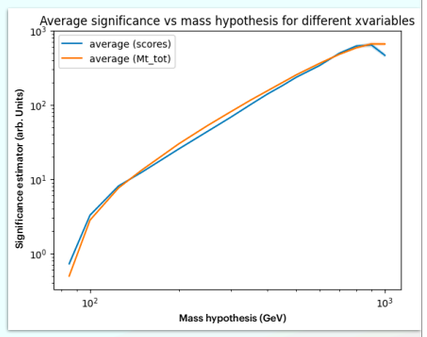
Model comparison

Significance plotting

- Standard Formula for Significance
  - Sig $= \frac{S}{\sqrt{B}}$ with custom bins
  - Sig. has arbitrary units
- Significance plot structure
  - X-axis: Mass Hypothesis
  - Y-axis: Average of significance scores



Increasing-Signal binning



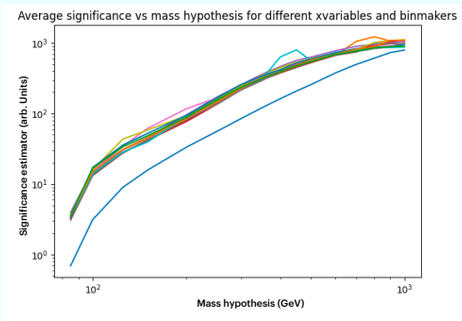
Constant Signal binning

To get a significance plot, we used the fairly standard S/sqrt(B) with the custom bin widths for each channel. Each histogram of a channel is essentially a point on the significance plot. The x axis is the mass hypothesis value and the y axis is the average of the significance scores of the 5 channels.

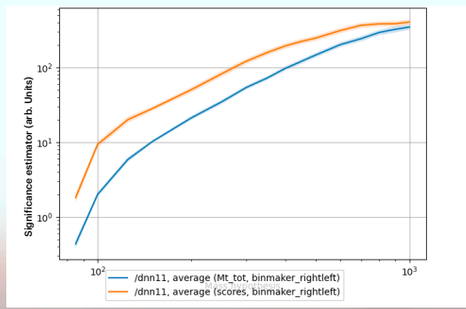
As you can see from the significance plots from both histogram functions the resulting plot looks similar. But since the increasing-signal histogram has more precision at high model score values, from now on I refer to only increasing-signal histogram significance plots. Note that this is using a recreation of Nelsons DNN model on pytorch with some minor differences.

DNN Training

- ✳ Initial goal
  - * Beat m_T^{tot} across all mass hypotheses
- ✳ Methodology
  - * Normalize inputs
  - * Try different depth and width combinations
- ✳ Findings
  - * Best model: 85 features, 2 layers [83, 30]



Comparing different models



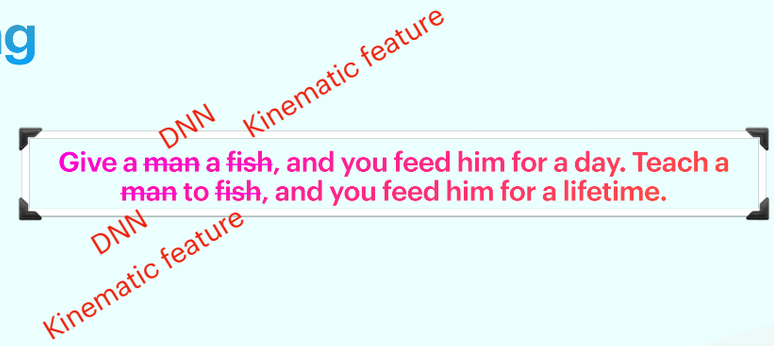
Best Model

Now that we have a way to compare models we can now start implementing different models to improve HNL classification. Our first goal was simple: make a model that can beat Mt_tot at all mass hypothesis since this must be possible as previously discussed. By normalizing the input features and running different model combinations we were able to beat this great foe *(im tired)*.

After testing around 40 models that had different widths, depths, binning functions, and input features we found the best model to take all 85 input features, and 2 hidden layers: [83,30]. As you can see they all perform similarly to each other and the improvement is only fractional. To improve the classification even more we need to change our approach to this classification.

Transfer Learning

Introduction



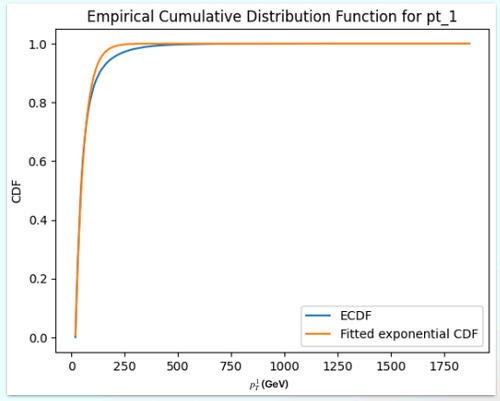
- Data Size Issue
- New Approach
  - **Transfer Learning**
- Regression DNN: predict calculated kinematic features
- Classification DNN: use regression model as input
- Advantages
  - Infinite synthetic event generation

Oftentimes this summer, our limiting factor has been data size. There is not enough data to be able to make complex ML structures without them overfitting. And so we tried a different approach: transfer learning. Except for the raw inputs, the calculated kinematic features can be, well, *calculated!* If we make a regression DNN that can predict accurately the calculated kinematic features we can use that model as the input to a classification DNN. This way it can use what it learned to calculate these features to improve its predictions. Furthermore, we are no longer restricted by the amount of data we have and we can generate an infinite amount of fake events for kinematic feature regression. Basically we have taken this famous quote to heart: Give a DNN a Kinematic feature, and you feed him for a day. Teach a DNN to kinematic feature, and you feed him for a lifetime.

Transfer Learning

Data Generation

- ✳ Additional output features
  - Mother particle kinematic values & E_{tot}
- ✳ Data cuts
  - Cut 0.03rd and 99.7th percentiles of real data
  - $\approx 27\,\%$ data removed
- ✳ Logical Limits
  - $\eta \in [-2.5, 2.5]$
  - $\phi \in [-\pi, \pi]$
  - p_T : exponential CDF



Fitted Exponential CDF

Firstly, we added more features that we though might help like the kinematic values of the mother particle and the total energy of the particles and MET. While we can create infinite amounts of events for the regression model, we still need to choose physically possible input features. To do when we calculate the output features to use as the labels we remove all events who's output features lie outside of the 0.03rd and 99.7th percentile of the real data. These cuts removed approximately 27% of the events. For input data we make logical limits such as eta between -2.5 and 2.5, and phi between -pi and pi etc. For Pt, however, we fitted an exponential CDF to approximate. While its not perfect its good enough for our purposes.

Transfer Learning

Regression Training

- ✳ Network
  - 1024 nodes
  - 25 layers
  - ~25M parameters
- ✳ Normalization
  - GeV vars divided by E_{tot}
- ✳ Training
  - New data every epoch
  - Loss function: MSE & Relative MSE
  - Optimizer: Adam + Decaying Learning Rate

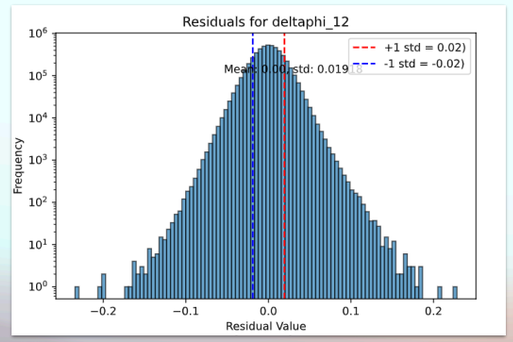
Best model validation loss

Now that we have the data generation set up, we can start training the regression model. With the additional kinematic features we have 112 output variables to regress. This is quite a lot and so we need a relatively big network to train. Because the GeV kinematic values can get large, we found that dividing them by E_tot helped the model to learn. This meant that all but E_tot itself would be limited by 5 and -5 or even tighter bounds. To not worry about overfitting, especially with such a large model, new data is generated before every epoch. This means that we can have many epochs of training. For example, our current best model trained on 1.6 billion events. This specific model has a width of 1024 nodes and a depth of 25 layers. This results in about 25 million trainable parameters. We used Mean square error (MSE) for all features except for E_tot where we used Relative MSE. We also used Adam with an exponentially decaying learning rate. An additional unorthodox thing we did was that with such a big network and some of the features

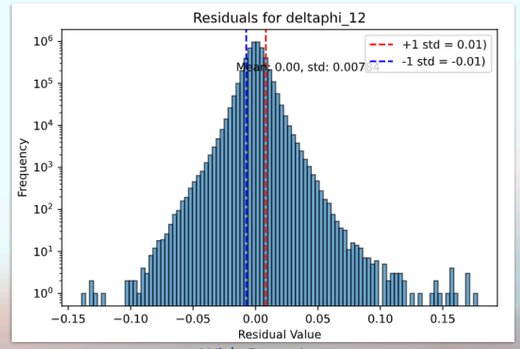
Transfer Learning

Drop-in technique

- Challenge
  - Losing input feature values
- Solution
  - “Drop-In”: Reintroduce inputs every 3 layers



Without Drop-Ins



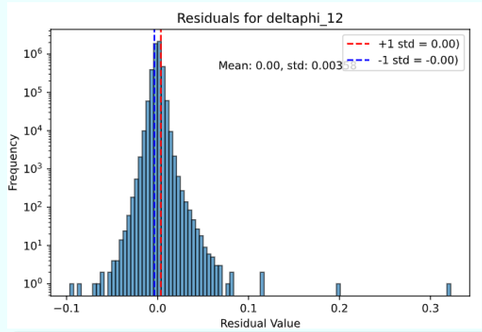
With Drop-Ins

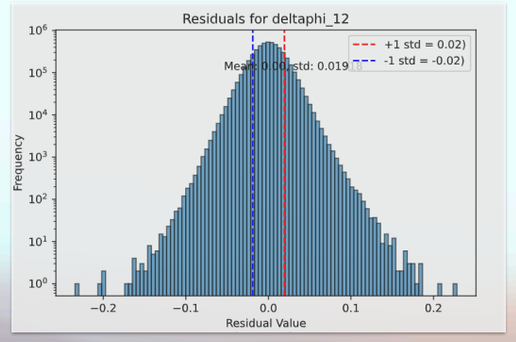
An additional unorthodox thing we did was that since some of the output features were quite simple mathematically like delta phi between different particles, we reintroduced the inputs every 3 layers to make sure the model did not forget these values by the later depths. Im not sure if theres a name for this technique but I refer to it as “drop-in”. Shown here on the left the residual of the deltaphi of between particles 1 and 2 without the drop ins and with the drop ins on the right.

Transfer Learning

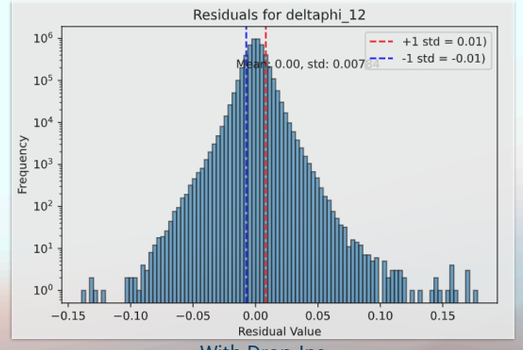
Drop-in technique

Best model:





Without Drop-Ins



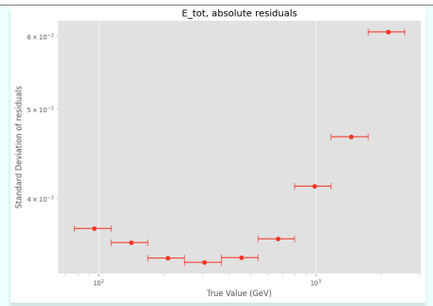
With Drop-Ins

An additional unorthodox thing we did was that since some of the output features were quite simple mathematically like delta phi between different particles, we reintroduced the inputs every 3 layers to make sure the model did not forget these values by the later depths. Im not sure if theres a name for this technique but I refer to it as “drop-in”. Shown here on the left the residual of the deltaphi of between particles 1 and 2 without the drop ins and with the drop ins on the right.

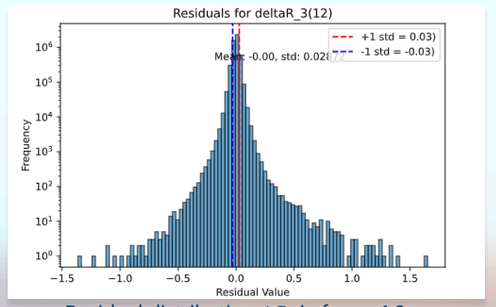
Transfer Learning

Regression Results

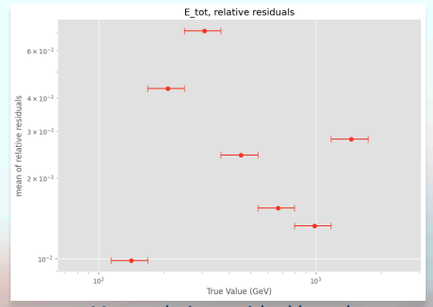
- ✳ Safety check
  - Make sure E_tot is being predicted well
- ✳ Challenges



Standard deviation bar plot of Absolute residual



Residual distribution ΔR_3 in frame 1,2



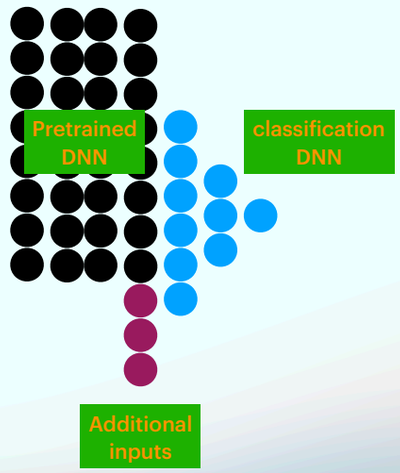
Mean relative residual bar plot

Because we divided each GeV feature by E_tot, it was crucial the the model could predict E_tot well. The plots here show how our best model performs at different E_tot true values.

Transfer Learning

Classification!

- Start point
  - Best multivariate regression model (pretrained)
- Data Prep
  - Remove last output of pretrained DNN
- Additional Inputs
  - Channel, Mass Hypothesis, particle charges
- Classification DNN
  - Depth 3



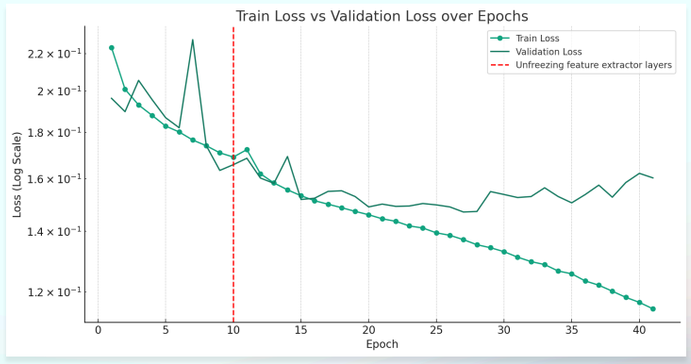
Visualization of Transfer Learning model

Now that we have a good model to predict kinematic feature values, lets move on to the main goal, classification. We did this in the following way, we load the best model and input the simulated “realer” data. We then run it through the model but remove the last “output layer” of the pretrained model. This means that we are left with a 1024 node latent representation of what the model finds useful to calculate the kinematic features. We the take this latent representation and combine it with other input data that is useful such as Channel, mass hypothesis, and particle charges. We then feed these into more DNN layers (which are not pretrained) of depth 3 and use a sigmoid activation in the end to classify.

Transfer Learning

Transfer Learning Strategies & Overfitting

- Options for transfer learning
  - Fixed weights
  - Unfrozen weights
  - Partially unfrozen weights
- Overfitting Risks
  - Escalates when weights are unfrozen
- Mitigation techniques
  - Adamw with L2 regularization
  - Dropout layers
  - Pruning



With transfer learning there are different options of how to use the model. The first is to not do backpropogation on the pretrained model which means that its weights do not change. This means only a the 3 depth classification model is trained. This means there are a lot less trainable parameters and overfitting is not as much of an issue.

The second option is to unfreeze the weights of the pretrained model from the beginning. However this leads to a massive amount of trainable parameters and the program overfits incredibly fast. The third option is the first freeze the pretrained models weights for some number of epochs and then unfreeze it.

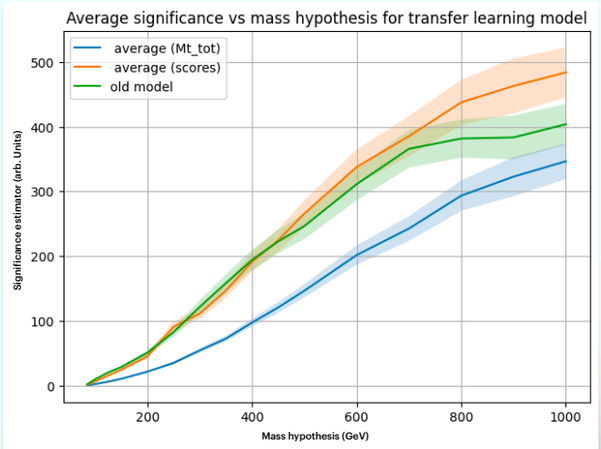
As shown in the figure, as soon as the weights are unfrozen, it takes only a few epochs for overfitting to occur.

We tried several ways to limit overfitting, we tried using Adamw activation function which has L2 regularization within it, we added drop out layers in the pretrained model when it was unfrozen, and we tried pruning where we remove weights from the multivariate regression model to have a smaller number of trainable parameters.

Transfer Learning

Model

- Pretrained model
  - 1024 width best model
  - Added dropout layers (50% chance)
- Whole model:
  - Unfrozen weights at epoch 5
  - Binary Cross Entropy loss
  - One-hot encoded channel input
  - Hidden layers: [128,128]

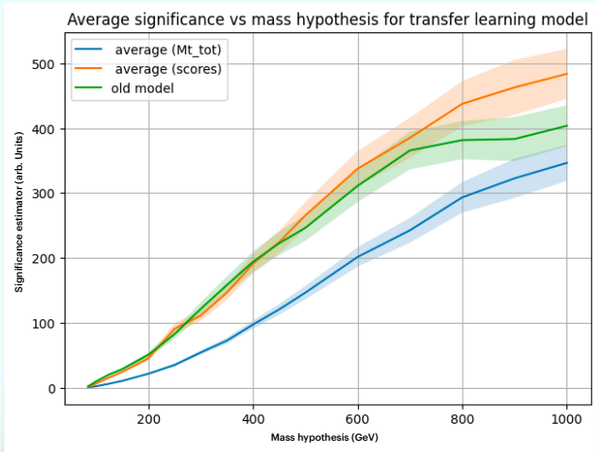


Significance estimator for Transfer learning model

Transfer Learning

Analysis

- Analysis
  - Model performs better than simple DNN at higher mass hypotheses
  - Similar to simple DNN at smaller values
- Possible Improvements
  - Try out other overfitting techniques
  - Stronger Regularization
  - Smaller regression network



Significance estimator for Transfer learning model

Now for what you've been waiting for, results! These are the significance plots of some of the best transfer learning models, the original simple DNN model, and Mt_tot.

Conclusion