Vision Transformer based Hadronic Tau Identification for the Dual-Readout Calorimeter

Youngwan Son, University of Seoul
On behalf of the Korea Dual-Readout Calorimeter R&D team
IDEA meeting, 9th Mar 2022
Introduction

This work aims to develop $\tau$ lepton identification based on dual-readout calorimeter image via deep learning, “Vision Transformer”.

* Investigate DRC standalone information potential for $\tau$ lepton identification.
* Vision Transformer (ViT) based classification of hadronic $\tau$ decays and QCD jets.

<table>
<thead>
<tr>
<th>Tau Branching ratio</th>
<th>$\pi^- \nu_\tau$</th>
<th>$\pi^- \pi^0 \nu_\tau$</th>
<th>$\pi^- 2\pi^0 \nu_\tau$</th>
<th>$\pi^- \pi^+ \pi^- \nu_\tau$</th>
<th>$\pi^- \pi^+ \pi^- \pi^0 \nu_\tau$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\tau$</td>
<td>10.91</td>
<td>25.51</td>
<td>9.29</td>
<td>9.00</td>
<td>2.70</td>
</tr>
</tbody>
</table>
WHY WE NEED DEEP LEARNING?
Universal Approximation Theorem: Neural Networks (NN) with Activation function (Nonpolynomial/Nonlinear function) can be universal approximated to any function like takes a number image as an input and tell the number as output.

-> That’s a strong guarantee for using NN, but the people didn’t know “how to approximate”, and that methodology is “Deep Learning”.

HOW TO APPROXIMATE?
1. Define loss function, which can represent a difference between model’s prediction about an input and wanted output.
2. Minimize loss function by gradient descent algorithm, which is an iterative optimization for finding the minimum of a function.
3. Back-propagate the error from the loss function to the Neural Network’s parameters by calculating each partial derivative of the loss function with respect to each parameter.
Simulate $e^+e^-$ beam at 91.2 GeV (Z boson mass) with decaying to $Z \to q\bar{q}$ or $Z \to \tau^-\tau^+$ with dual-readout v0.0.1 simulation software.

* Options : $|\eta| < 1.15$ for final state particles, No magnetic field applied.
* Five dominant hadronic decays of $\tau^\pm$ are identification target with QCD jets from $Z \to q\bar{q}$.

Cluster and generate 256x256 images about each type of fibers.

* Geometry : $(\Delta \theta, \Delta \phi) = (0.5,0.5)$ with respect to the center of jet cluster
* Pixel information : Energy of Cherenkov and Scintillation channel
* A pixel is not one-to-one corresponded to a fiber. Just fill 256x256 ROOT::TH2F of the $(\Delta \theta, \Delta \phi) = (0.5,0.5)$ region.
Transformer network is originated from sequential data processing without RNN (“Attention is all you need”).

**Vision Transformer (ViT)** takes sequential patches of image as input, and it learns the relations between each patches.

Transformer based models are beating up Convolutional models for many vision tasks. (i.e. **Image classification**, **Object detection**)

Is there any ViT application in HEP? -> Nothing (yet).
Patch Embedding block
Key feature >> einops.layers.keras.Rearrange('b (h s1) (w s2) c -> b (h w) (s1 s2 c)', s1=patch_size, s2=patch_size)
* Map 2D image ($x_p \in \mathbb{R}^{H\times W\times C}$) to Patch embedding ($x_p \in \mathbb{R}^{N\times(P^2\times C)}$).
* i.e.) Height (H) = h*s1 = 256, Width (W) = w*s2 = 256, Channel (C, Cherenkov&Scintillation) = 2, patch_size(P) = 32
  * Number of patches (N) = HW/P^2 = 64 (Same with upper image of patches)
* After this layer, tensorflow.keras.layers.Dense (Linear projection) takes the patch embedding and map to linear embedding.

Minor Features : They are implemented by tensorflow.Variable().
* [CLS] token : It is introduced in BERT paper for classification task of transformer. It learns aggregate representation of a image.
* Position embedding : Patch embedding loses its spatial information. It learns original positions of the patches.
**Tau Identification - Vision Transformer**

**Transformer Encoder block**
Repeat this block L times, the L is given by developer.
Key feature >> `tf.keras.layers.MultiHeadAttention(num_heads=8, key_dim=emb_size, ...)`
* Do multi-head **self-attention** to the embedding (Patch Embedding block output).
* After this layer, MLP takes the output and upsample it to higher dimension and downsample to the original dimension (emb_size).

Attention map visualization -> Attention layer knows where to pay attention!

**Minor features**:
* **GELU**: $GELU(x) = x\Phi(x)$ where Gaussian CDF $\Phi(x)$ -> Differentiable for every point, not monotone increasing.
* **Layer Normalization**: Normalization with stability for small batch size -> enables smoother gradients, etc.
**Classification Head**

Key feature >> tensorflow.keras.layers.Dense(n_classes, activation='softmax')

* It takes reduced tensor (reduce $P^2C + 1$ dimension by taking mean along to the axis.), and map to the probability vector.

* For the probability vector, each elements mean the probability of each decays.

  * i.e.) $[\pi^+\nu_\tau, \pi^+\pi^+\pi^+\nu_\tau, \pi^+\pi^+\pi^+\pi^0\nu_\tau, \pi^+\pi^0\nu_\tau, \pi^+2\pi^0\nu_\tau, q\bar{q}] \rightarrow [0.95, 0.01, 0.01, 0.01, 0.01, 0.01]$  

  * The prediction is $\pi^+\nu_\tau$. 

---

**Tau Identification - Vision Transformer**

![Diagram of Vision Transformer](image)
Hyperparameter optimization was done by optuna, which is based on tree-structured Parzen estimator.

Two step tuning
1. Model structure
   * depth (L) : The number of repetition of transformer layer
   * emb_size : Size of embedding
   * patch_size : Size of rectangular patches for (256, 256) image
   * The best hyper parameter set for 100 trials of 20 epochs
      * (depth, emb_size, patch_size) = (1, 640, 16)

2. Hyperparameters for training
   * batch_size : Input tensor shape -> (batch_size, 256, 256, 2)
   * lr : Learning rate, step size at each iteration while moving to the minimum of loss function
   * The best hyper parameter set for 100 trials of 20 epochs
      * (batch_size, lr) = (60, 0.001041)
Hyperparameter optimized model

Patch Embedding -> patch_size=16, emb_size=640
-> 257 = Number of patches + [CLS] Token = (256/16)^2 + 1

Transformer -> depth (=L)=1
-> MLP upsamples the attention output (4 times), and then downsamples to original dimension. -> 2560 = 640 × 4

Classification head

Loss is simple categorical cross-entropy. Adam optimizer is used for gradient descent. Train with 260,000 (image, label)
* Test with 19,920 images, equipartite with each decay modes.
* Each row of confusion matrix represents actual class, and each column of confusion matrix represents predicted class.
* $Z \rightarrow q\bar{q}$, main background of hadronic $\tau$ identification is identified truly more than 99.5%.
* Average accuracy of the identifier is more than 97.7%.
Summary & Plan

* Achieve nice performance of $\tau$ lepton identification task with standalone (DRC info. only) by Vision Transformer model.
  * 97.8% (total) accuracy for the 6-class classification of QCD jets and five hadronic $\tau$ decays.
  * Can discriminate QCD jets with 99.5% accuracy.

* We have a plan to apply this for Physics case like H-$\rightarrow$$\tau^-\tau^+$ in FCC-ee Higgs boson (ZH) production.
* Attention map visualization may give us intuition and interpretability, so I’m working in.
Back up
1. The cartoon shows how to calculate the attention value about the sentence “I am a student” by matrix multiplication.

2. Self-attention calculates the similarities between each word, so it finds “it” is highly associated with “animal”.

The cartoons show how to calculate the attention value about the sentence “I am a student” by matrix multiplication. Self-attention calculates the similarities between each word, so it finds “it” is highly associated with “animal”.