Project No: B3127-211

NANYANG TECHNOLOGICAL UNIVERSITY

# $PVT^3$: A Pruned Video-Vision Transformer for Tactile Texture Classification


**Submitted by: Ouyang Yanjia**

**Matriculation Number: U1822883C**

**Supervisor: Prof. Lin Zhiping**

**Co-supervisor: Dr. Wu Yan(A*STAR-I2R)**

School of Electrical & Electronic Engineering


A final year project report presented to the Nanyang Technological University

in partial fulfilment of the requirements of the degree of

Bachelor of Engineering

**2022**

# Table of Contents

# Abstract

With the newly involved technologies in tactile sensory, variants tactile sensors have been deployed on robots which provides them touching ability to perceive complex environments. One typical example of robot touching task is to recognize different materials based on the tactile data generated from different textures. In this report, we propose $PVT^3$, a light-weight Transformer-based architecture with pruning layers to model the texture representation. By using a Video-Vision Transformer backbone, the spatial and temporal features will be well preserved and utilized. The multi-dimensional pruning layers will reduce model complexity and size without sacrificing the performance. Three tactile datasets are used for testing the $PVT^3$ model. Overall, our proposed model achieves higher accuracy on material classification results with a smaller model size compared to the state-of-the-art tactile texture models.

This work was written as a paper and submitted to the International Conference on Intelligent Robots and Systems (IROS) 2022.

# Acknowledgements

It is a great journey for me to research on this Final Year Project with the help of my dear supervisors and friends. I feel very grateful for their guidance and support during my university life.

First, I would like to thank my supervisor Prof. Lin Zhiping who encourages me to challenge myself and push me towards a better version. He plays a vital role in this project by sharing his knowledge and expertise during meetings and phone calls.

Second, I am thankful for my co-supervisor Dr. Wu who are always there to clear my doubts and show me the right direction. It is of great pleasure to learn from him about how to do research.

Third, my senior Ms. Gao Ruihan gives me lots of support even though there is a twelve-hour time difference between us. I was impressed with her composure in her research and her positive outlook on life.

Finally, I would like to express my thankfulness to my families and friends for their care and companionship. Without them, I would not have such a wonderful memory in my university life.

Ouyang Yanjia

March 2022

# Acronyms

| | |
|---|---|
| CNN | Convolutional Neural Network |
| MLP | Multi-Layer Perceptron |
| RNN | Recurrent Neural Network |
| LSTM | Long-Short Term Memory |
| GRU | Gated Recurrent Unit |
| ViT | Vision Transformer |
| ViViT | Video Vision Transformer |

# List of Figures

# List of Tables

# Chapter 1
# Introduction

## 1.1 Motivations

Robots are used in variance environments to play different roles. For an unstructured environment such as a crowded shopping mall or wild nature reserve, there are many undefined dynamics or messages which are difficult for robots or even human beings to notice and understand depending only on a single sense. Despite the vision and audition which are easy to acquire, touching is also an important way to collect useful information as a supplement. This sensing modality works well in tasks requires low latency such as recognizing material properties [1] [2] , texture of surfaces [3] [4], objects shapes [5] [6] [7] and controlling the interaction between objects [8] [9].

Unlike vision and audition which use conventional sensors and have generally accepted standards, tactile sensing has more diversity since it can interact with objects in many ways. To adapt to different tasks, extensive research has been taken to develop new tactile sensor technologies [10] [11] [12]. Some biologically inspired tactile sensors mimics human fingers [13], some use special materials to extract surface texture with high resolution [11]. Although the sensors have different implementations, when applied for collecting tactile data, most of them still follows strict rules about the contacting time, force, torque, or other control protocols. This not only requires delicate operation but is very different from how we touch objects in our daily lives. For example, when we want to feel the material of clothes, we may use our fingers or

palms to move in whatever directions with different forces. There is no need to keep our fingers moving at a constant speed or along a single line. We have a great deal of freedom when touching. Intuitively, when touching an object, it is not the touching methods but the properties of the object itself matters. Therefore, this suggests that it should have a strong self-attention property between the tactile sensory data and the object characteristics.

Extensive research projects in tactile region have explored a variety of machine learning algorithms [14] [15] [16]. Tactile sensory data is different from vision and audition for its own spatial-temporal representation. To tackle this complex correlation in both space and time domain, previous works focus on modelling both spatial and temporal parts individually and integrate the two parts into a holistic learning framework. Those proposed frameworks all achieve high accuracy in tactile based tasks which implies the two-stage model is suitable for spatial-temporal data. For spatial component, Convolutional Neural Network (CNN) or Multi-Layer Perceptron (MLP) is used for extracting important features [17]. For temporal components, Recurrent Neural Network (RNN) is the basic idea to modelling sequential data. However, RNN has a drawback. Since it takes the input one by one and use previous output as input for next step, it suffers from gradient vanishing and slow training. Therefore, when facing long sequence data, RNN cannot fully use the entire information which lowers its performance.

To compensate the unsatisfactory performance, advanced frameworks are used such as Long-Short Term Memory (LSTM) [15] [18] and Gated Recurrent Unit (GRU) [19]. Although these advanced frameworks solve tactile based classification problem with very high accuracy, it is worth to notice that those RNN based spatial feature selection

models still require high computation power and consumes lots of time when facing long sequence data.

To overcome this disadvantage, we introduce a transformer-based architecture which has a parallel pipeline that can significantly reduce the difficult in dealing with long sequence data. Transformer structure has been proved to be useful in both Natural Language Processing (NLP) [20] and Computer Vision (CV) [21] [22]fields. As discussed before, tactile based data contains both spatial and temporal information. Transformer is designed to handle sequential data just like tactile data. And according to our daily life experience, the spatial-temporal representation of tactile data should have some level of self-correlation. Therefore, utilizing a transformer-based model for analysing tactile data for material classification task should have a good result.

## 1.2 Objectives and Scope

The main contribution of this report includes:

- Propose a pruned transformer-based model $PVT^3$ with a ViViT [21] as the backbone for classifying different materials by using texture representation of tactile data
- Apply the $PVT^3$ model into three different datasets each with different sensors and methods of data collection to prove the generalization capability of the model
- Process a new dataset collected using GelSight [11] sensor for 45 materials, which can be a supplement for texture classification learning in the future

In addition to the three contributions mentioned above, the work presented in this report is also submitted to the International Conference on Intelligent Robots and Systems (IROS) 2022 under review.

This report aims at applying state-of-the-art model structure to robot tactile sensing tasks and is mainly focusing on material classification based on texture representation provided by different sensors.

## 1.3 Organizations

The structure of this report is designed as follows. Chapter 2 gives a comprehensive literature review about the transformer development to give the reader a better background knowledge. There typical transformer model will be presented as they show the steps to fit the model for spatial-temporal data which is closely related with our research problem. Information about different sensors and their collected data will also be introduced in this chapter. Besides, we will also mention pruning techniques in this chapter as our final model includes some pruning layers. Chapter 3 presents the model $PVT^3$ and the training steps in details including preprocessing for different datasets. Chapter 4 focus on experiment results, and we use previous RNN-based model as a benchmark to show the improvements of our new model. Finally, Chapter 5 is the conclusion part and future research directions.

# Chapter 2
# Literature Review

Tactile based sensory data provides useful information in both space and time domain. This complex feature needs a special designed model structure to resolve the underneath information. For different tactile sensors, they may need different methods to collect texture data and the collected data may vary in shapes and sizes. In this chapter, we will first introduce the development of transformer architecture which achieve very well performance in lots of different tasks and should have the potential to solve material classification task based on texture information as well. Then, we will introduce the three sensors which corresponding to the three datasets we use in the experiment. Lastly, since we want to further improve the model, we will give a review on model pruning techniques.

## 2.1 Transformer

Transformer is first published in 2017 and then becomes very popular in lots of machine learning related fields such as natural language processing [20], audio processing [23] and computer vision [21] [22]. The original Transformer [20] is built for language translation problems which has an encoder-decoder structure with attention mechanism to analysis the global correlations in sentences. A typical model structure diagram is shown in figure 1 [20].

The input (e.g., English sentences) will first go through an embedding block to be transformed into a set of vectors and then be put into the encoder block. The output

(e.g., German translation) will also be embedded and put into the decoder block. Before putting into the encoder or decoder block, inputs and outputs get a positional encoding which will add a number based on the sequence of original words of the sentence to represent the original location. This step directly tells the model the original location of different words and therefore the words in the sentence do not need to be put into model one by one to keep the sequence information but can be put together and realize a parallel computation.

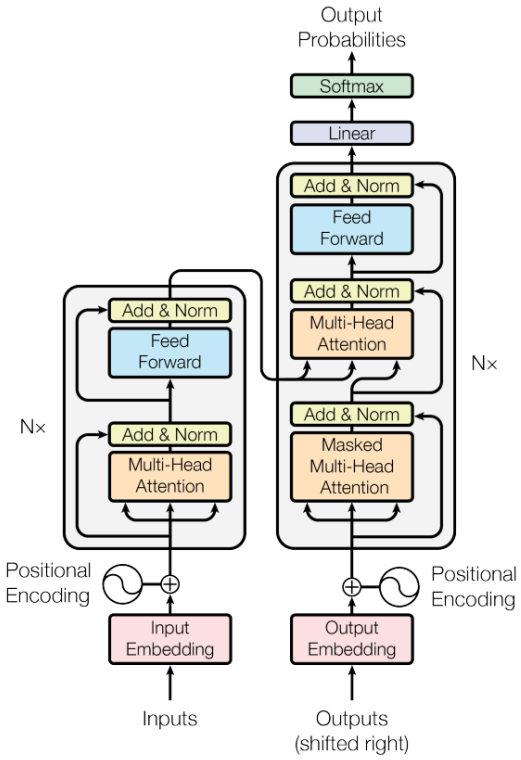


*Figure 1: Transformer model structure [20]*

Attention function is the most important feature for the Transformer structure. This function creates three vectors which are used to calculate the scores between different words within a sentence. The three vectors are named as Query, Key and Value (QKV). By creating three QKV matrices, each embedded word matrix will multiply with QKV matrices to create their own Query, Key and Value set. Each set will be used to compute

the attention score with all sets including itself. This computation is called “Scaled Dot-Product Attention”.

From figure 1, there is no block named attention, but some blocks called Multi-Head Attention. That is because the Transformer structure uses multi-set of QKV matrices, with each named as a “head”. By assumption, different heads focus on different aspects of information, therefore some heads may have a good understanding for close context while another head may focus on relation between two words far away from each other. In total, 8 heads are used to construct the Transformer model, and the size of QKV matrix is reduced to lower the computational cost.

However, although Multi-Head Attention structure achieves good result for this language translation task, later researchers [24] proposed that it is not necessary for the high performance of the model. Reducing the number of heads forces the remaining heads to focus on more aspects and takes on more features.

Multi-Head Attention layer essentially is just linear transformation since it only using multiplication and addition. To introduce nonlinear transformation into the model, Feed Forward Network is used which includes fully connection layers, dropout layers and ReLU activation function. These two sub-layers creates the main body of a Transformer block. In details, a residual connection and normalization layer is added for each sub-layer, creating a more powerful model.

The duplication of encoder and decoder blocks stacked together creates a deeper transformer network which aims to have better capacity in understanding complex information.

## 2.2 Vision Transformer

Vision Transformer (ViT) [22] inherits the original schema from Transformer model but applied for image classification jobs and achieves competitive results with conventional CNN structure.

The innovative part of ViT [22] is that it first introduces transformer structure for 2-dimensional datatype. A typical size for image classification tasks is 256*256 pixels. In order to adapt the images to the new model structure and keep the good parallel effect of transformer, preprocessing of images imitates the steps in NLP. That is, for a square image, it will firstly be split into 16 small patches with each size 16*16. Then, those patches will be flattened into 1 dimensional array and be embedded through a fully connection layer. A position indicator token and a classification token are added onto each embedded array. After all these steps, 16 vectors are put into the transformer encoder block to experience further computation.

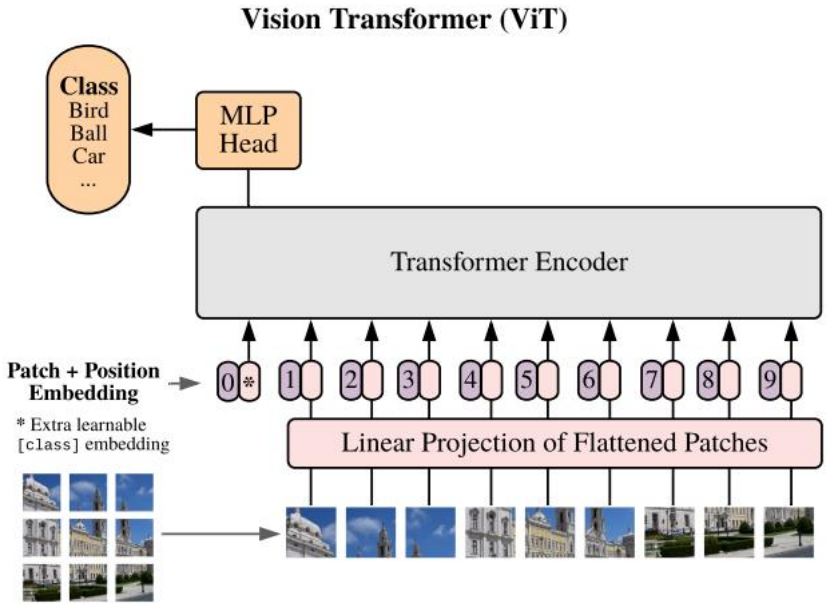


*Figure 2: ViT model structure [22]*

One thing that noticeable is the classification token called "CLS token". This token is widely used in classification tasks using transformer structure [22] [25]. Since most transformer-based model uses Multi-Head Attention to compute, too many features become an obstacle in deciding the final output. The proposed CLS token is used as the only factor for final classification prediction layer, while it gets update in training and absorb all the useful information from heads.

Another difference compared to original transformer structure is that ViT [22] uses learnable position embedding. Compared to the manual designed position embedding, this design is also able to keep the position information after splitting.

One problem for transformer-based image classification mentioned in [22] is the low performance in tasks with small amount of data. For large scale image datasets such as ImageNet [26] and CIFAR-100 [27], ViT achieves state of the art classification

accuracy. But for small dataset such as medical images [28], the performance is not satisfactory. The transformer structure limits the ability for the model to grasp local information like CNN. In contrary, ViT utilize global information very well. Therefore, feed in enough data will help ViT to gain the same or better ability in understanding images. However, this not only need a huge dataset but also requires a powerful GPU or TPU to train the model. Since it is expensive and time consuming to train a pure transformer-based model for computer vision tasks, many variants of ViT is proposed [29] [30]. By including an extra feature selection block, these models [31] [32] take advantages from both CNN and transformer structure. The hybrid architecture enhances the model capability and able to use a small parameter size.

## 2.3 Video Transformer

As the name indicates, video transformer is applying transformer structure for videos such as video recognition and video classification. Video can be viewed as a sequence of images, just like the tactile dataset we have, videos is also a spatial-temporal representation that needs to consider both domains. Compared to our tactile dataset, videos are often larger in size. It has more time frames, and each frame size usually is not small. For example, Kinetic400 dataset [33] has videos around 10 seconds with each frame size about 256*256.

Video is a 3-dimensional data, it can put into transformer model directly if divided into small cubes. Two models are implemented in work [34] and [21] although the results are fine, it can be further improved by using two separate parts. One part is used to analysis the spatial dynamics and the other part is to model the temporal representations. For such large input size, extracting important features separately in

space and time domain rather than considering the video as a whole can significantly reduce the cost. [35] and [21] both proposed such kind of model and named as Video Transformer Network (VTN) and Video Vision Transformer (ViViT). VTN [35] tried lots of spatial backbones including ViT, ResNet [36], DeiT [37], and use Longformer [38] as its temporal encoder. ViViT [21] use pure transformer structure for both spatial and temporal backbone but modified a bit to create 3 different video classification models.

## 2.4 Tactile Sensor

Tactile information is difficult to define as its format depends on tactile sensor and data collection method. Many efforts have been made to design and develop new type of tactile sensors which helps robots to perceive the environment accurately. Popular approaches in tactile data collection include three technologies, which are based on soft robotics, biomimetics and optical sensors [12]. Although the implementation for tactile sensors may be different, those sensors can be used together as well. For instance, Li [7] utilizes multiple sensors to record pressure force, object thermal conductivity, object temperature and environment temperature to represent tactile information. Moreover, tactile data collected using different sensors for material classification can be used together by joint learning to achieve higher classification accuracy [18].Three distinct tactile datasets are used in this project. They are collected from three types of sensors. SynTouch BioTac [13] sensor use biomimetic technique made from rubber-like material which mimics human finger. It is able to feel the pressure, vibration and temperature. iCub RoboSKIN [39] is an implementation of soft robotics which is basically a pressure sensor. iCub is cost-effective compared to BioTac but gives out more noise. The last sensor used in this project is GelSight [11]

which is a high-resolution optical tactile sensor which uses elastomer to record the convexity of texture.

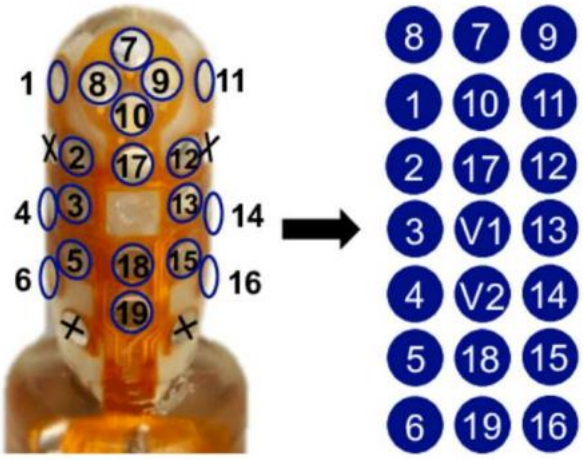


*Figure 3: Schematic diagram of the BioTac [13]*

*Figure 4: pressure-sensitive artificial skin of the iCub [40].*

## 2.5 Pruning Technique

Model compression technique reduce model size and remains model performance.

Since transformer is often criticized for its large parameter size, applying pruning onto transformer model should decrease its model size and save memory and computation costs. Common model compression techniques include quantization [41] [42], knowledge distillation [43] and model pruning [44] [45] [46] [24]. Details for the three compression methods will be provided in the following sections.

Quantization utilizes matrix factorization, vector factorization or other algorithms to lower the bit width when storing tensors. It finds a round value for the original number which occupies less memory space. Choosing a relatively small bit width will significantly reduce the model size but may influence the model performance. If a bit width smaller than 8 bits are chosen, the final memory will be reduced by 4 times or higher [42]. Binary quantization will make all the weight equal to 0 or 1. Besides quantization by bit width, there are also algorithms such as k-means quantization which use clustering to divide original parameters into a few groups. All the numbers inside the same group will be assigned a group index, and the group centroid value will be stored in a look up table. When using the model, the index will be replaced by the group centroid value.

Knowledge distillation is a powerful tool that can make small model have almost the same performance with very big model. The pretrained large model will act as a teacher to guide the small model serves as student to learn. Large transformer model needs lots of data and time to training and remains complex when applied to test cases. A small model to replace the large model will get trained efficiently by absorbing the information provided by teacher network and the trained student network can be implemented for tasks with limited memory space and computing power environment. A typical example [25] is using Bidirectional Encoder Representations from

Transformers (BERT) to train a small network to do intent classification.

Pruning techniques introduced by Le Cun et al. [47] remove connections between neurons to reduce the model complexity. Once the pruning criteria is selected, the model will be compressed and retrained repeatedly to reduce size while remaining model performance. Pruning can be categorized into two types. First type is unstructured pruning usually prunes out more parameter weights but needs hardware implementation to utilize the sparsity created by pruning to accelerate the computation [48]. Second type is structured pruning which changes the model structure to reduce model size. For CNN models, pruning can be achieved in four levels [49]which are weight-level, kernel-level, channel-level, and layer-level. Inspired by them, we can apply those pruning strategies to make the transformer-based architectures more efficient. Weight-level pruning is often unstructured therefore easy and flexible. As mentioned before, unstructured pruning is hard to speed up training unless special implementation has been done in software or hardware [48]. Kernel-level and channel-level pruning are both structured pruning method, but the pruning happens in different positions. In transformer, Multi-Head Attention layer and Feed Forward Network layer consume most of the computing powers. Pruning out excessive heads, patches or dimensions should be a good idea to compress the model [44] [45] [46]. Layer-level pruning [50] does not require any further change in software or hardware implementation but is often used for very deep neural network. Although transformer model can go very deep, for this specific task, we did not implement a deep neural network therefore we did not consider it in following pruning parts.

# Chapter 3
# Methodology

We will first give a short description about the backbone structure of video transformer and then introduce the pruning algorithms we used to reduce the model parameter size. The model structure diagram is shown in figure 5.

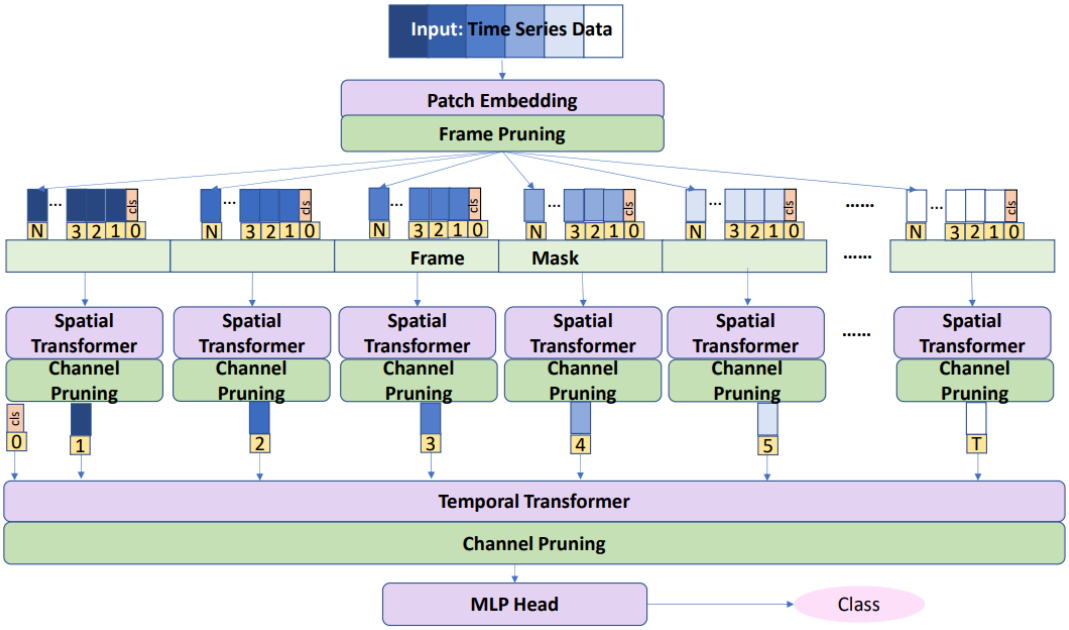


*Figure 5: $PVT^3$ model structure*

## 3.1 Backbone

The backbone structure we choose is a two-transformer-stacked structure which is borrowed from ViViT [21]. Since tactile information is hidden in the interaction or contact events between sensor and materials, the pure attention structure in ViViT should be able to catch the important correlations between the changes of these spatial-temporal signals. The Factorized Encoder model presented in [21] shows very high performance in video tasks with fewer floating-point operations (FLOPs). And the

two-transformer-stacked structure is similar to the existing tactile analysis approaches [15] [19] which intuitively process the tactile data step by step to better understand the information in space domain and time domain. This structure also limits the computing cost as it leaves out unnecessary connections between some time domain information and space domain information and just to include the most important parts.

In ViViT backbone, two transformers are stacked together. Each transformer closely follows the vision transformer structure. The time series data will go through the first transformer named spatial transformer which only considers attention score between patches comes from the same time stamp. The next transformer, temporal transformer will receive the output from spatial transformer and relates all the frames to determine the final prediction result.

In details, tactile data will be loaded as a vector in the form of [b h w t] where b, h, w, t stands for batch size, input height, input weight and input time step. Patch embedding layer will embed input into an embedded vector [b t n e] where n, e stands for patch number and embedded size. Then, a classification token is concatenated to the embedded vector create vector [b t (n+1) e]. As the first spatial transformer only considers space domain, column t is merged into column b, leaving only the last two column to be used for attention calculation. A learnable position embedding vector will be added to include the original position information before the Multi-Head Self Attention (MHSA) and Feed Forward Network (FNN) layers. After the process in spatial transformer, only the classification token value will be kept and passed to the next transformer so the input to the temporal transformer has shape [b t e]. Similar process is performed to create a vector with shape [b (t+1) e] where the concatenated vector is the classification token for temporal transformer. Temporal position

embedding is added to the vector to represent the time sequence. Finally, the classification token whose value is from attention scores of the temporal transformer will be used to give out the classification prediction in the last Multi-Layer Perceptron (MLP) block.

As discussed in literature review, the Multi-Head Self Attention (MHSA) and Feed Forward Network (FNN) layers contains most of the parameters. Two transformers will create two sets of MHSA and FFN with the same depth for each transformer. Therefore, the total parameters needed for ViViT is much larger than ViT. However, it can be shown that the computing effort needed for ViViT is much fewer [21]. And ViViT can achieve better result than simply linking all time frames and using ViT to do classification. More details will be provided in Chapter 4.

## 3.2 Pruning

Although ViViT saves lots of computing efforts, it is still reasonable to explore more about the pruning to further decrease model size and find the appropriate hyperparameters needed to train the model with high accuracy and low cost. Here, we present two types of pruning. One is to use the built-in pruning functions in Pytorch to do the post-training pruning. The other is to modify the model and write some pruning layers to cut some connections inside the model. We will briefly talk about the post-training pruning and focus more on the second training method.

Post-training is a very simple way to compress the model. As we have access to all the trained parameters, it is easy to make some of those equal to zero and finish the training.

However, this method needs to know every layer name to tell the program where to find the parameters which is not very user-friendly for people who are not familiar with the inner structure. We just use this method to prove that the model can be compressed but we do not care about why to prune those parameters as we are using a random pruning in our experiments. More precisely, we use this random pruning as a function who plays a role like dropout to make the model more generalizable to achieve higher accuracy in test data.

The other pruning method is structure pruning. Two kinds of such block are added, channel selection and frame selection. Channel selection is inserted at the MHSA and FFN layers as these two layers contains most of the parameters therefore it may have lots of redundancy. Frame selection is inserted after the patch embedding layer which reduce the input at the beginning.

The pruning layers are implemented by masking out unimportant features. We initialize the mask as a matrix filled with one. This matrix is trainable and will change its value with the training steps. The value in this matrix is considered as the pruning score while higher value means the corresponding parameters are important and lower value means the corresponding parameters can be pruned out without highly affect the model performance. When pruning, a threshold will be set which is a percentage tells the program how many percent we want to prune the parameters. According to this percentage, the program will calculate the threshold where all values below this threshold will be set to zero and the corresponding features will have no effect on the final classification output.

1) Channel selection

In Multi-Layer Self Attention (MLSA) block, three matrices will be created. These matrices will transpose input into Q K V which stand for Query, Key and Value. Attention score is calculated using these numbers by equation:

$$\text{Attention}(\mathrm{Q}, \mathrm{K}, \mathrm{V}) = \text{Softmax}\left(\mathrm{QK}^{\mathrm{T}}/\sqrt{d}\right)\mathrm{V} \quad [22]$$

Channel selection will prune out some values in Q K V transpose matrix for both spatial and temporal transformers. This step can reduce the numbers for attention score calculation. Also, we apply channel selection to fully connection layers in Feed Forward Network (FFN) block to create more sparsity. As these two blocks contain so many parameters, pruning a large portion will not affect the overall accuracy.

2) Frame selection

As the tactile data contains so many time frames, depending on different sensor frequency and interaction duration, we actually do not have to use all of them. Experiment in [15] proves reducing time frames in a range do not really affect the total model performance, using only a fraction of total data can also learn the tactile representation with high accuracy. This experiment leads us to think if we can reduce the input frame form the beginning of the model and we implement a frame selection layer to leave out the redundancy in time domain. This helps to significantly reduce the temporal features passed to the model. The exact proportion of pruning how many time frames depends on the complexity of the textures present, sensor frequency and the recorded duration. If the texture present contains many noises or they are very similar from each other, we should only remain most of the time frames. If the sensor frequency is high and recorded duration is long, we can prune out more time frames as the input is still abundant.

## 3.3 Baseline

Both the ViT [22] and the ViViT [21] are used as benchmark for iCub dataset and BioTac dataset. The details are provided in Chapter 4 Section 4.1 and 4.2. We use recurrent autoencoder (RAEC) model from [18] as the baseline model to compare with the $PVT^3$ for all three datasets. Chapter 4 Section 4.3 and 4.4 will talk more about the experiments. When choosing the model hyper-parameters for $PVT^3$, we set similar parameter size with RAEC which is about 0.1 million trainable parameters.

# Chapter 4

# Experiments and Results

This chapter shows the datasets we used and the training and pruning results in detail.

## 4.1 iCub Dataset

This dataset [15] contains information of 20 different materials with 50 samples for each class. The tactile data are captured by iCub RoboSKIN tactile sensor when the iCub robot sweeps its forearm in rotation. Each sample has a time length of 75 and the 60 electrodes are processed into 6 × 10 matrix.

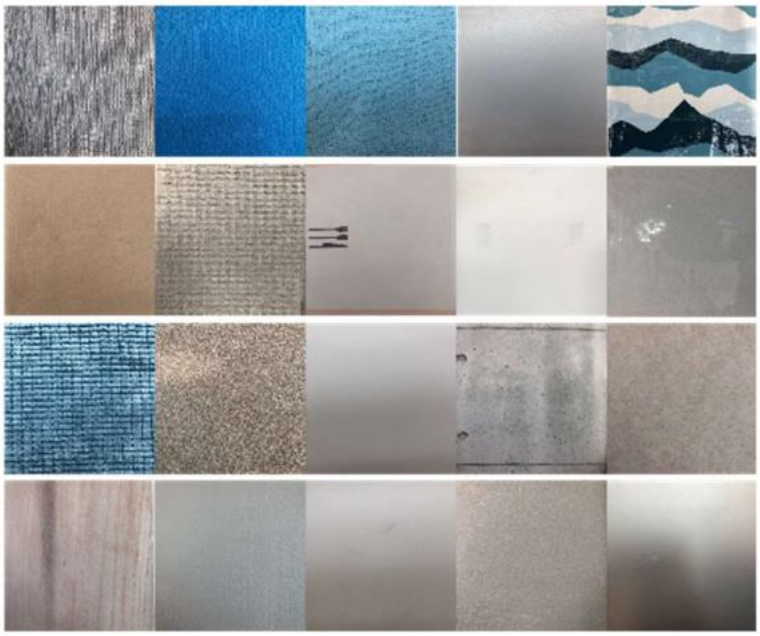

*Figure 6: 20 materials*

This dataset is used for both one-transformer structure (ViT) and two-transformer structure (ViViT). For ViT, different time frames are attached together to form a big frame. For ViViT, the input is passed as a time sequence. The parameter size we used for ViT is larger than ViViT. The learning curves are shown in below. It is obvious that the ViViT model achieves better training accuracy and validation accuracy and the variance in the last 100 epochs is smaller which suggests the ViViT model is more stable.

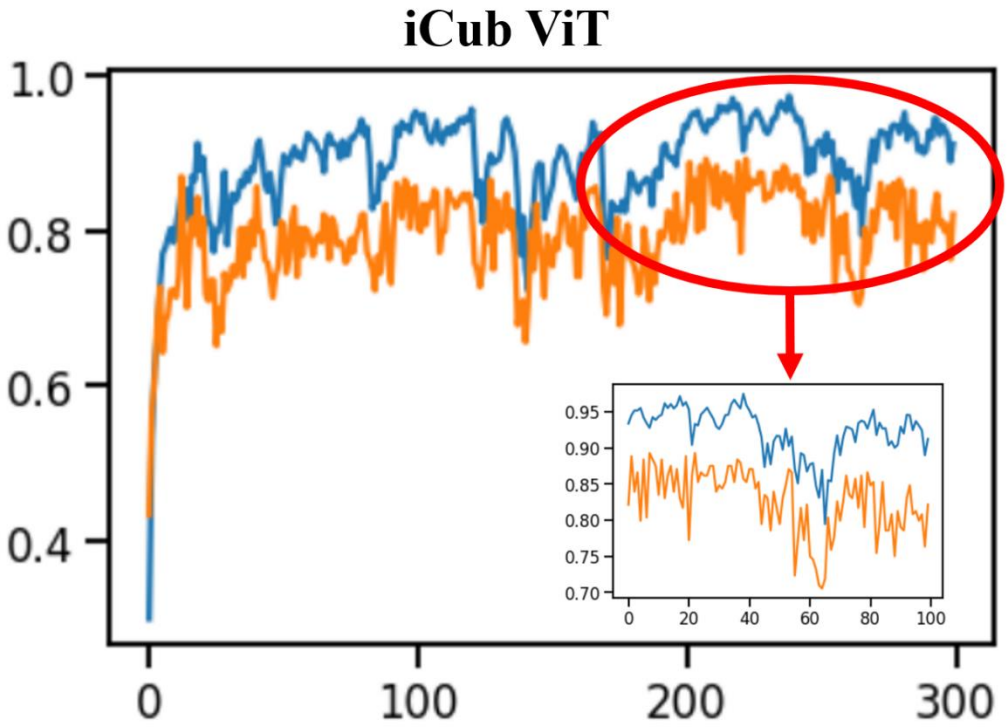


*Figure 7: iCub ViT best validation accuracy 0.8884*

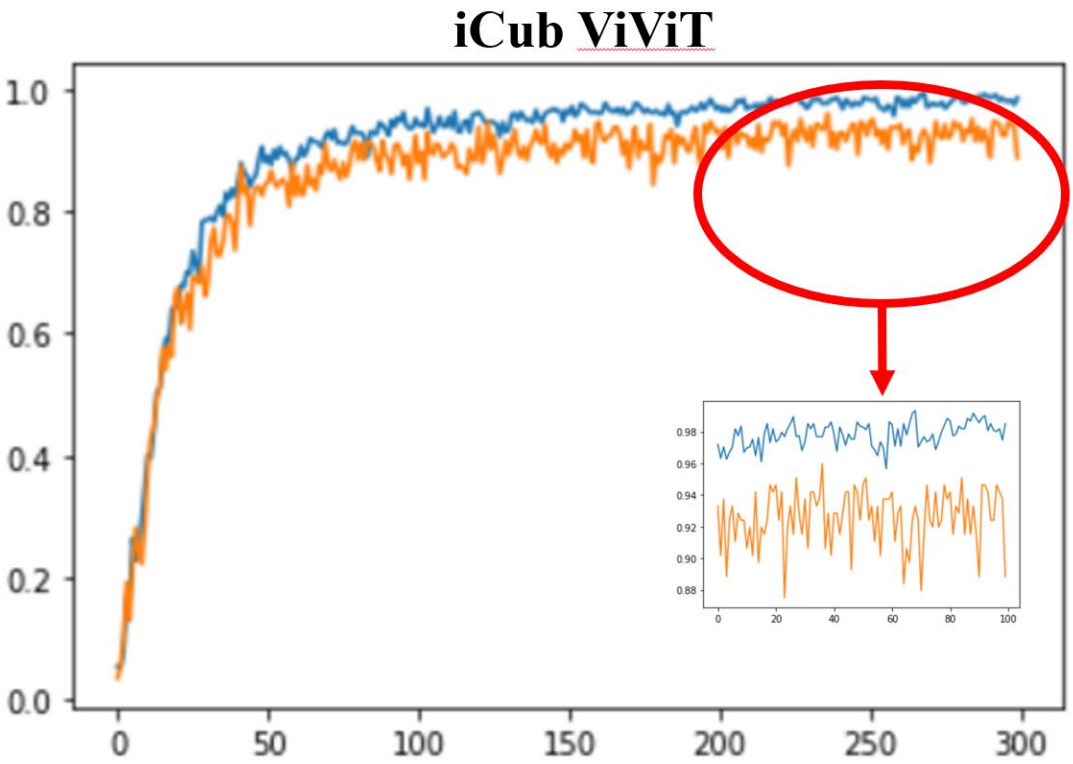


*Figure 8: iCub ViViT best validation accuracy 0.9598*

## 4.2 BioTac Dataset

This dataset [18] [19] contains two parts. One part collects 20 materials as same as the iCub dataset. The other part collects 50 materials shown in figure 9. The data is collected in sliding motion using SynTouch BioTac tactile sensor who has 19 electrodes.

This dataset is also used for comparing the ViT and ViViT model with same model hyper-parameters as the iCub dataset used. The training curves are shown in below. Compared to iCub dataset, BioTac dataset has less noise and is simpler to recognize

different materials. Therefore, for both models, BioTac accuracy is higher than the iCub accuracy.

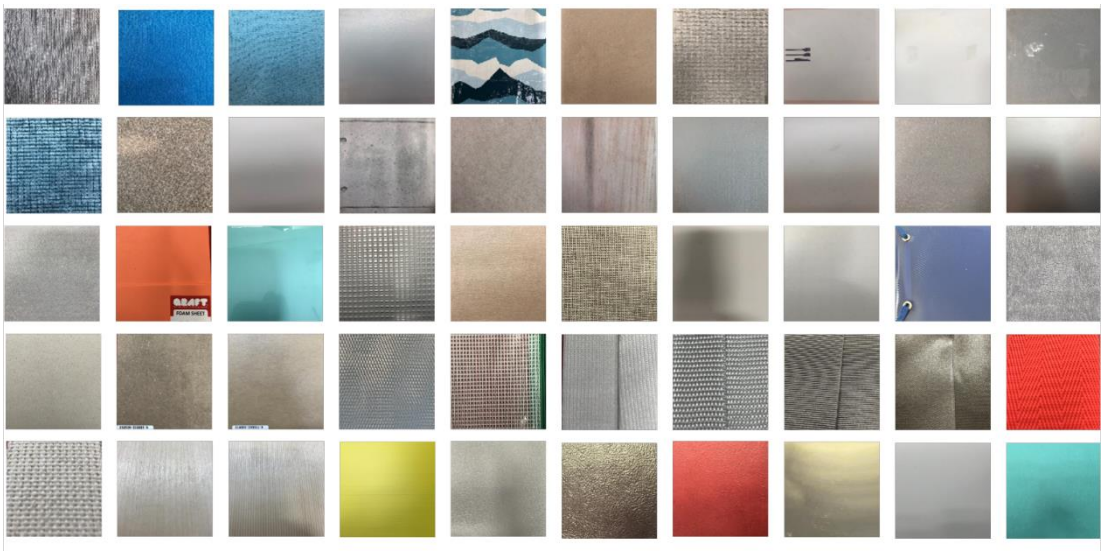

*Figure 9: 50 materials*

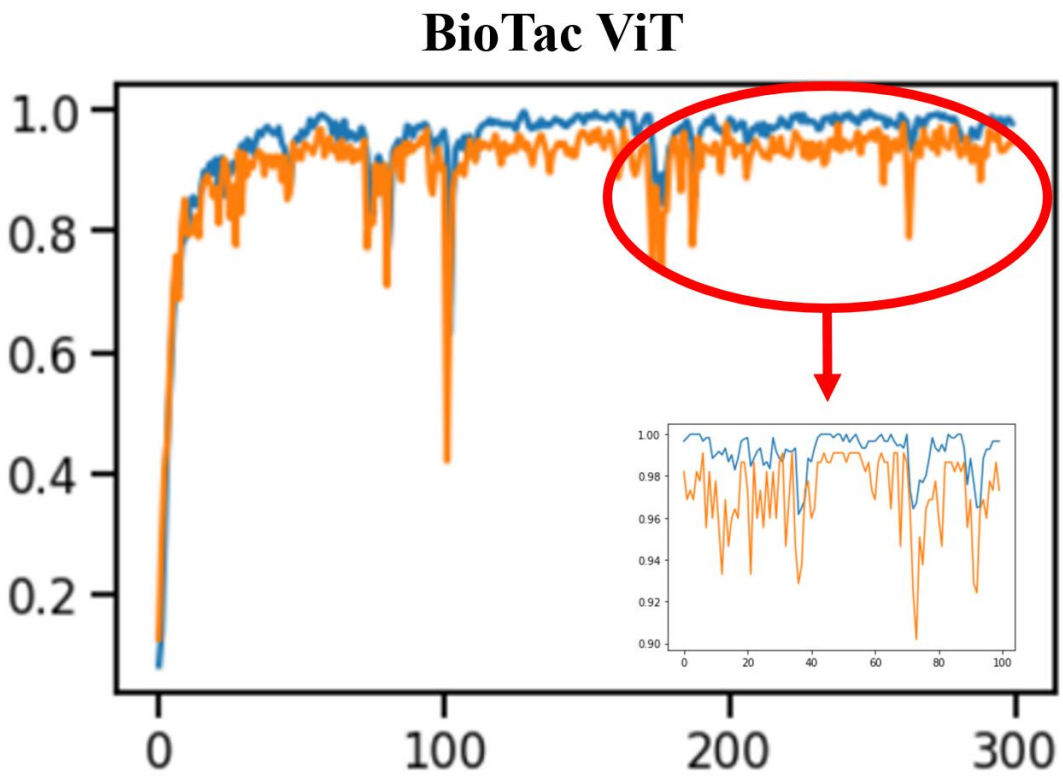


*Figure 10: BioTac ViT best validation accuracy 0.9777*

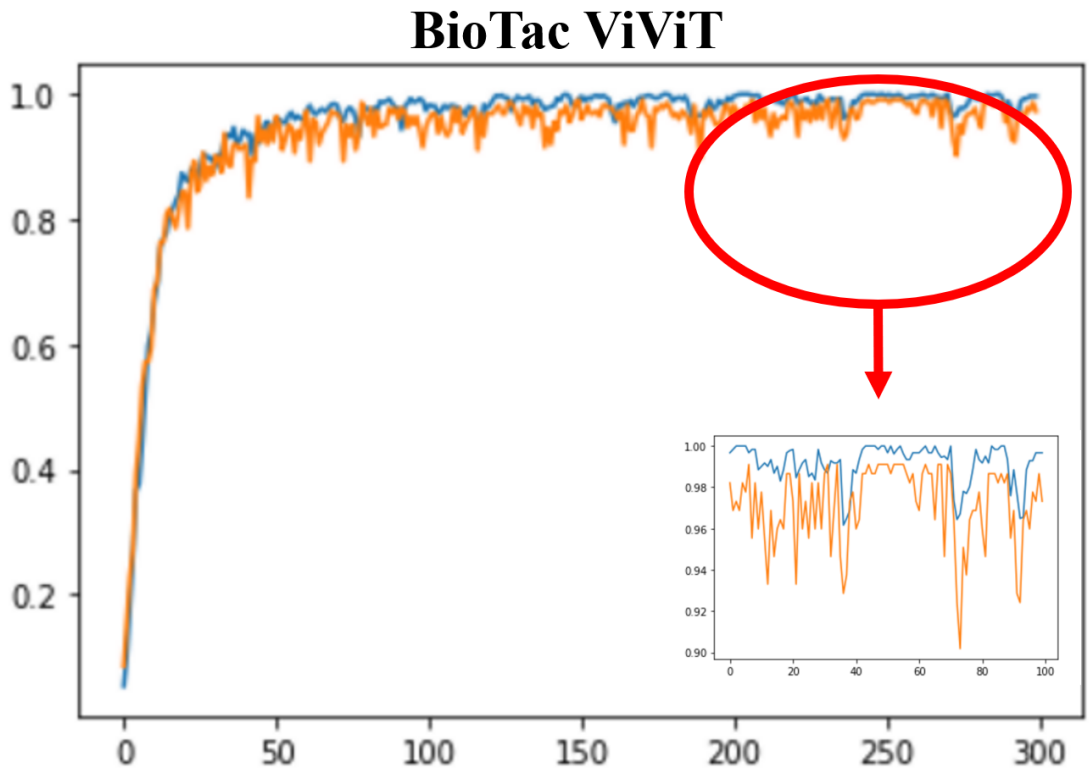


*Figure 11: BioTac ViViT best validation accuracy 0.9911*

## 4.3 GelSight Dataset

This dataset is collected from GelSight sensor which is a pad mode from elastomer. Since the pad is vulnerable and fraction between pad and material will cause damage to the pad surface, the data is collected by rolling the sensor on the materials instead of sliding on them. A total number of 45 materials with each 50 samples are available as a dataset, and 15 kinds of materials are used in this experiment. The GelSight will create a high resolution RBG image for each sample represent the tactile information. The images for each material are shown in figure 12.

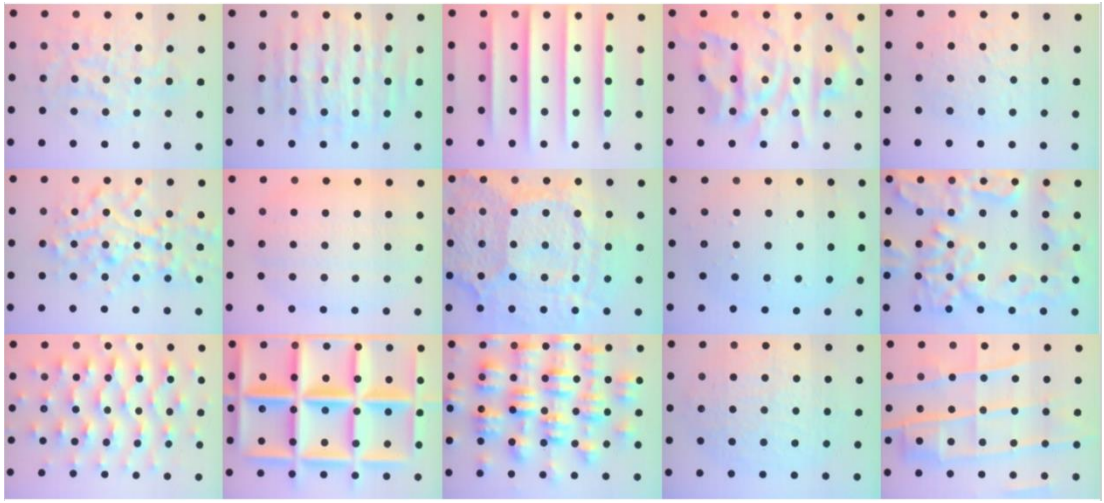

*Figure 12: 15 materials*

As the GelSight creates very high-resolution images, the size of this dataset is very large. Although we only test the model on one third of the total dataset, it is still very change for our computing resource to handle such a large amount of data. Therefore, the first thing for this dataset is to do some preprocessing jobs to limit the data size. Originally, each time frame size is of 480 × 640 pixels, and each sample contains about 300 frames. Not all the time frames contain useful information because there is sometimes the sensor do not contact with the materials, but the sensor is keeping generate readings. According to the csv file which summarize the time stamp about at what time the sensor starts or ends the interaction with materials, we select the time frames that the gel pad contact with material with maximum contact surface and reduce the time frame to 25 frames. Also, we crop the original image into a 240 × 320 pixels size to further reduce the data amount.

## 4.4 Training Result

*Table 1: Comparison of test accuracy for 4-fold cross validation*

| Average Test Accuracy | | |
|---|---|---|
| Dataset / Model | iCub | BioTac20 |
| RAEC (independent) | 0.9114 | 0.9531 |
| RAEC (joint learning) | 0.9323 | 0.9601 |
| **$PVT^3$ without pruning** | **0.9224** | **0.9659** |
| **$PVT^3$ with RANDOM pruning** | **0.9425** | **0.9798** |

For $PVT^3$, the model is trained for 300 epochs for each k-fold cross validation set. And each training is repeated 5 times and the trained models are tested on test set creating 5 different test accuracies. The final accuracy is the average of the 20 test accuracies for each 4 folds and 5 times.

As compared to RAEC [18] with independent training, transformer-based $PVT^3$ show better performance and the increase in accuracy is about 1.2%. but compared to RAEC [18] with joint learning, $PVT^3$ is better on the BioTac20 but gets lower accuracy for iCub dataset. For all the models, BioTac20 dataset gets higher accuracy result than iCub dataset, this outcome is in line with the observation in [18] that BioTac dataset is much cleaner and is easier to differentiate materials from the tactile representations.

*Table 2: Performance of ViViT backbone on BioTac dataset*

| BioTac 20 and BioTac 50 Classes | |
|---|---|
| Dataset | Average Test Accuracy |
| BioTac 20 | 0.966 |
| BioTac 50 | 0.947 |
| Combined | 0.903 |

The BioTac 20 and BioTac 50 shares similar tactile representation for the common 20 materials since the collect method and sensor are the same. However, due to experiment setup or sensor wear, the collected data may be different in scaling and bias. The model performance for both BioTac 20 and BioTac 50 are very high, however, the accuracy for the combined dataset is not as high as the individuals. This suggests the bias or scaling factor can confuse the model. But the accuracy is still above 90% which means the transformer-based model maintains a reasonably robust performance in spite of the distribution.

For GelSight dataset, the data is too big for long pipeline architecture models such as REAC due to the weak supervisory signal at the end of the pipeline. It may needs longer training time or to manually select important features first to help REAC get better result on this dataset. A background subtraction process is applied to the raw GelSight dataset to normalize the data before putting into models. The testing result are 89.3% and 16.0% accuracy for ViViT backbone and RAEC model respectively. The testing result shows ViViT can handle inputs with large number of features by integrates the attention for both space and time domain, but the RAEC model is

difficult to extract important features when giving too many features.

## 4.5 Pruning Result

*Table 3: Channel selection results*

| Channel Selection Pruning | | |
|---|---|---|
| Dataset | Test Accuracy before Pruning | Test Accuracy after Pruning |
| iCub fold 1 | 0.925 | 0.940 |
| iCub fold 2 | 0.935 | 0.940 |
| iCub fold 3 | 0.945 | 0.945 |
| iCub fold 4 | 0.935 | 0.950 |
| BioTac fold 1 | 0.975 | 0.965 |
| BioTac fold 2 | 0.970 | 0.980 |
| BioTac fold 3 | 0.965 | 0.965 |
| BioTac fold 4 | 0.980 | 0.975 |

Using iCub dataset as an example, which originally have a sample shape of H ×W ×T = 6×10×75, where 75 is the length of time frames. For this dataset, $PVT^3$ has a total number of 64272 trainable parameters. Inputs are split into 4 patches and each patch is embedded into a 1-D tensor of length 20. Concatenate classification token will create a [5×20] shape tensor to feed in spatial transformer. The Channel Pruning layer in spatial transformer will force some of values in Q K V transpose matrix equal to zero. Each transpose matrix is of size [5×20×64]. The mask in Channel Pruning layer will

prune out values in the last column create new matrix with size [5×20×(64*p)] where p is the pruning percent.

In temporal transformer block, whose input is [76×20], where 76 comes from the 75 time frames plus 1 temporal classification token, the Q K V transpose matrix size will change to [76×20×64]. The mask is put on the first column in temporal transformer.

Other than the two, Channel Pruning is also applied to FFN block where some connections in the fully connection layer will be pruned out.

*Table 4: Frame pruning with different ratio*

| Frame Selection Pruning | | |
|---|---|---|
| Pruned Rate | BioTac20 | iCub |
| 30 percent | 0.970 | 0.915 |
| 40 percent | 0.960 | 0.895 |
| 50 percent | 0.945 | 0.650 |
| 60 percent | 0.940 | 0.545 |
| 70 percent | 0.930 | 0.630 |
| 80 percent | 0.895 | \ |

The full data for BioTac is 400 time frames, and the full data for iCub is 75 time frames. In real time, BioTac collection lasts for 8 seconds while the iCub collection only lasts for 3 seconds. We want to reduce the used inputs to mimic if the contact time between

tactile sensor and material surface is short, which can be further modified to do real-time inference. Experiment in [15] shows that selecting only half of the time frames can achieve relatively good performance. Therefore, we add a frame selection layer after patch embedding layer to reduce the used inputs by transformers. Table 4 summarizes the results.

For both datasets, pruning lowers the model performance. For BioTac, the effect is minor, and the classification accuracy is still high after pruning 70 percent of the original time frames. However, for iCub dataset, the classification accuracy drops quickly even if we do not prune over 50 percent.

Possible reasons for the drop in accuracy are:

1) The data we used is collected from only sliding mode, however, data used in [15] utilize both dynamic touch and sliding motion which contains richer information.
2) The change between each frame may be the most important information to reveal the tactile representation, however, since we are not select a continuous portion of the dataset, we may lose this important feature if too many frames are pruned out.
3) iCub dataset has a smaller temporal resolution at the beginning, therefore same pruning portion means less remained time frames. Therefore, iCub suffer a sharper drop in classification accuracy compared to BioTac dataset.

# Chapter5
# Conclusion and Future Work

In this chapter, we will summary the main contribution of this report and propose the future improvement directions for following research works.

## 5.1 Conclusion

This final year project studies about Transformer and its variants and presented the $PVT^3$ $model$ based on ViViT backbone with channel pruning in different layers to classify materials based on tactile sensory data acquiring from three different types of sensors. As compared to previous studies, $PVT^3$ achieves better results in classification accuracy and model size. We have also demonstrated the generatability of this model as it handles three different datasets of tactile based material classification tasks with high performance without specifically preprocessing for a variety of data size. We believe that our model should be useful for analyzing other tactile perception tasks as well since the end-to-end structure is easy for transfer learning. Last but not least, the pruning layers added before different blocks can give a good reduction in parameter size since they create lots of sparsity without sacrificing the overall performance with a proper pruning ratio. The pruning layer also serves as a tool to estimate the minimal number of parameters needed for different tasks and can be used as a baseline for compressing model further.

## 5.2 Future Work

We have demonstrated the high potential of transformer in tactile classification tasks, but we did not consider the original space representation of tactile data. For irregular inputs such as BioTac dataset, we just flatten it as a 1-D array other than the original triangular shape. Future work can explore about the patch size selection for different input size or utilize the original shape to achieve better model performance.

Other than that, transforming transformer-based model into spiking neural network (SNN) can faster the inference speed while maintaining high model performance. Since SNN is event-driven, it should be suitable to analyze the changes between tactile time frames and find the relation between those signals.

# Reflection on Learning Outcome Attainment

This Final Year Project is a challenging and rewarding process. I not only learned a lot of knowledge, but also mastered a lot of learning and research methods.

First of all, this project is a completely new field for me. This is my first experience with tactile sensors, and I don't have any experience with RNNs or transformers before. Although I did not collect data in the laboratory due to the epidemic, I also had a preliminary understanding of the Robot Operating System (ROS). In this project, I learned more about deep learning. I tried running the code in different environments and for the first time worked with a huge dataset (GelSight Dataset). Thanks to Dr. Wu and A*STAR for their help in enabling me to use a computer with more memory and computing power to do this deep learning work.

Secondly, this project has greatly improved my problem-solving ability. During the project, I encountered a lot of algorithms or codes that I did not understand. By reading the related paper and searching for forum discussions, I gradually cleared most of my doubts. Meanwhile, I have a new understanding of some knowledge points that I think I have mastered. I carefully analyse every change in the code to see how I can get better model results. During this process, my supervisors Prof. Lin and Dr. Wu gave me a lot of suggestions and told a lot of directions that I could research on. My seniors Gao and Tian are nice and patient that provided me with helpful documents. Gao helped me understand and process the tactile datasets and gave me valuable advice.

Last but not the least, the experience of FYP is very precious to me. I have met many

friends who are interested in scientific research at A*STAR. I am inspired by their enthusiasm for scientific research and serious attitude towards projects. I will embark on a new journey with the various research skills and knowledge I have acquired from FYP.

# References

[1] S. Luo, W. Yuan, E. Adelson, A. G. Cohn and R. Fuentes, "ViTac: Feature Sharing Between Vision and Tactile Sensing for Cloth Texture Recognition," *Intelligent Robots and Systems (IROS) 2018 IEEE/RSJ International Conference,* pp. 2722-2727, 2018.

[2] T. Taunyazov, L. S. Song, E. Lim, H. H. See, D. Lee, B. C. Tee and H. Soh, "Extended Tactile Perception: Vibration Sensing through Tools and Grasped Objects," *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS),* pp. 1755-1762, 2021.

[3] C. W. Fox, B. Mitchinson, M. J. Pearson, A. G. Pipe and T. J. Prescott, "Contact type dependency of texture classification in a whiskered mobile robot," *Autonomous Robots,* vol. 26, pp. 223-239, 2009.

[4] N. Jamali and C. Sammut, "Material classification by tactile sensing using surface textures," *2010 IEEE International Conference on,* pp. 2336-2341, 2010.

[5] M. Kaboli, A. D. L. Rosa T, R. Walker and G. Cheng, "In-hand object recognition via texture properties with robotic hands, artificial skin, and novel tactile descriptors," *2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids),* pp. 1155-1160, 2015.

[6] H. Liu, Y. Yu, F. Sun and J. Gu, "Visual-Tactile Fusion for Object Recognition," *IEEE Transactions on Automation Science and Engineering,* vol. 14, no. 2, pp. 1-13, 2016.

[7] G. Li, S. Liu, L. Wang and R. Zhu, "Skin-inspired quadruple tactile sensors

integrated on a robot hand enable object recognition," *Science Robotics,* vol. 5, no. 49, 2020.

[8] J. Bimbo, S. Luo, K. Althoefer and H. Liu, "In-Hand Object Pose Estimation Using Covariance-Based Tactile To Geometry Matching," *IEEE Robotics and Automation Letters,* vol. 1, no. 1, pp. 570-577, 2016.

[9] J. W. James, N. Pestell and N. F. Lepora, "Slip Detection With a Biomimetic Tactile Sensor," *IEEE Robotics and Automation Letters,* vol. 3, no. 4, pp. 3340-3346, 2018.

[10] N. Wettels, V. J. Santos, R. S. Johansson and G. E. Loeb, "Biomimetic Tactile Sensor Array," *Advanced Robotics,* vol. 22, no. 8, pp. 829-849, 2008.

[11] W. Yuan, S. Dong and E. H. Adelson, "GelSight: High-Resolution Robot Tactile Sensors for Estimating Geometry and Force," *Sensors,* vol. 17, no. 12, 2017.

[12] N. F. Lepora, "Soft Biomimetic Optical Tactile Sensing with the TacTip: A Review," *IEEE Sensors,* vol. 21, no. 19, pp. 121-143, 2021.

[13] Y. Chebotar, K. Hausman, Z. Su, S. G. Sukhatme and S. Schaal, "Self-Supervised Regrasping using Spatio-Temporal Tactile Features and Reinforcement Learning," *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS),* 2016.

[14] J. A. Fishel and G. E. Loeb, "Bayesian exploration for intelligent identification of textures," *Frontiers in neurorobotics,* vol. 6, p. 4, 2012.

[15] T. Taunyazov, H. F. Koh, Y. Wu, C. Cai and H. Soh, "Towards Effective Tactile Identification of Textures using a Hybrid Touch Approach," *2019 International Conference on Robotics and Automation (ICRA),* pp. 4269-4275, 2019.

[16] T. Taunyazov, W. Sng, B. Lim, H. H. See, J. Kuan, A. F. Ansari, B. C. K. Tee and H. Soh, "Event-Driven Visual-Tactile Sensing and Learning for Robots," *arXiv preprint arXiv:2009.07083,,* 2020.

[17] Y. Gao, L. A. Hendricks, K. J. Kuchenbecker and T. Darrel, "Deep Learning for Tactile Understanding From Visual and Haptic Data," *2016 IEEE International Conference on Robotics and Automation (ICRA),* pp. 536-543, 2016.

[18] R. Gao, T. Taunyazov, Z. Lin and Y. Wu, "Supervised Autoencoder Joint Learning on Heterogeneous Tactile Sensory Data: Improving Material Classification Performance," *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS),* pp. 10907-10913, 2020.

[19] R. Gao, T. Tian, Z. Lin and Y. Wu, "On Explainability and Sensor-Adaptability of a Robot Tactile Texture Representation Using a Two-Stage Recurrent Networks," *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS),* pp. 1296-1303, 2021.

[20] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser and I. Polosukhin, "Attention is all you need," *arXiv,* 2017.

[21] A. Arnab, M. Dehghani, G. Heigold, C. Sun, M. Lučić and C. Schmid, "ViViT: A Video Vision Transformer," *arXiv,* 2021.

[22] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit and N. Houlsby, "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale," *arXiv,* 2020.

[23] L. Dong, S. Xu and B. Xu, "Speech-Transformer: A No-Recurrence Sequence-to-Sequence Model for Speech Recognition," *2018 IEEE International*

*Conference on Acoustics, Speech and Signal Processing (ICASSP),* pp. 5884-5888, 2018.

[24] P. Michel, O. Levy and G. Neubig, "Are Sixteen Heads Really Better than One?," *arXiv,* 2019.

[25] J. Devlin, M.-W. Chang, K. Lee and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," *arXiv,* 2018.

[26] J. a. D. W. a. S. R. a. L. L.-J. a. K. L. a. L. F.-F. Deng, "ImageNet: A large-scale hierarchical image database," *2009 IEEE Conference on Computer Vision and Pattern Recognition,* pp. 248-255, 2009.

[27] A. Krizhevsky, "Learning Multiple Layers of Features from Tiny Images," 2009.

[28] C. Matsoukas, J. F. Haslum, M. Söderberg and K. Smith, "Is it Time to Replace CNNs with Transformers for Medical Images?," *arXiv,* 2021.

[29] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin and B. Guo, "Swin Transformer: Hierarchical Vision Transformer using Shifted Windows," *arXiv,* 2021.

[30] K. Han, A. Xiao, E. Wu, J. Guo, C. Xu and Y. Wang, "Transformer in Transformer," *arXiv,* 2021.

[31] H. Wu, B. Xiao, N. Codella, M. Liu, X. Dai, L. Yuan and L. Zhang, "CvT: Introducing Convolutions to Vision Transformers," *arXiv,* 2021.

[32] S. a. T. H. a. L. M. a. M. A. a. B. G. a. S. L. d'Ascoli, "ConViT: Improving Vision Transformers with Soft Convolutional Inductive Biases," *arXiv,* 2021.

[33] W. Kay, J. Carreira, K. Simonyan, B. Zhang, C. Hillier, S. Vijayanarasimhan, F. Viola, T. Green, T. Back, P. Natsev, M. Suleyman and A. Zisserman, "The

Kinetics Human Action Video Dataset," *arXiv,* 2017.

[34] Z. Liu, J. Ning, Y. Cao, Y. Wei, Z. Zhang, S. Lin and H. Hu, "Video Swin Transformer," *arXiv,* 2021.

[35] D. Neimark, O. Bar, M. Zohar and D. Asselmann, "Video Transformer Network," *arXiv,* 2021.

[36] K. He, X. Zhang, S. Ren and J. Sun, "Deep Residual Learning for Image Recognition," *arXiv,* 2015.

[37] H. Touvron, M. Cord, M. Douze, F. Massa, A. Sablayrolles and H. Jégou, "Training data-efficient image transformers & distillation through attention," *arXiv,* 2020.

[38] I. Beltagy, M. E. Peters and A. Cohan, "Longformer: The Long-Document Transformer," *arXiv,* 2020.

[39] A. Schmitz, M. Maggiali, L. Natale, B. Bonino and G. Metta, "A tactile sensor for the fingertips of the humanoid robot iCub," *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems,* pp. 2212-2217, 2010.

[40] A. Roncone, M. Hoffmann, U. Pattacini and G. Metta, "Learning peripersonal space representation through artificial skin for avoidance and reaching with whole body surface," *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS),* 2015.

[41] G. Klein, Y. Kim, Y. Deng, J. Crego, J. Senellart and A. M. Rush, "OpenNMT: Open-source Toolkit for Neural Machine Translation," *arXiv,* 2017.

[42] A. a. S. V. Bhandare, D. Karkada, V. Menon, S. Choi, K. Datta and V. Saletore, "Efficient 8-Bit Quantization of Transformer Neural Machine Language

Translation Model," *arXiv,* 2019.

[43] D. a. H. K. Jia, Y. Wang, Y. Tang, J. Guo, C. Zhang and D. Tao, "Efficient Vision Transformers via Fine-Grained Manifold Distillation," *arXiv,* 2021.

[44] M. Zhu, Y. Tang and K. Han, "Vision Transformer Pruning," *arXiv,* 2021.

[45] Y. Tang, K. a. W. Y. Han, C. Xu, J. Guo, C. Xu and D. Tao, "Patch Slimming for Efficient Vision Transformers," *arXiv,* 2021.

[46] F. Lagunas, E. Charlaix, V. Sanh and A. M. Rush, "Block Pruning For Faster Transformers," *arXiv,* 2021.

[47] Y. L. Cun, J. S. Denker and S. A. Solla, "Optimal Brain Damage," *San Francisco, CA, USA: Morgan Kaufman Publishers Inc.,* pp. 598-605, 1990.

[48] S. Han, X. Liu, H. Mao, J. Pu, A. Pedram, M. A. Horowitz and W. J. Dally, "EIE: Efficient Inference Engine on Compressed Deep Neural Network," *arXiv,* 2016.

[49] Z. Liu, J. Li, Z. Shen, G. Huang, S. Yan and C. Zhang, "Learning Efficient Convolutional Networks through Network Slimming," 2017.

[50] G. Huang, Y. Sun, Z. Liu, D. Sedra and K. Weinberger, "Deep Networks with Stochastic Depth," 2016.

# Appendix

In this appendix, we present code snippets for GelSight preprocessing.

In [2]:
```
import pandas as pd
import numpy as np
import cv2
import os
```

In [6]:
```
    i=1
    dirc = "A:/FYP/data/GelSight/GelSight/"
    f = open(str(dirc)+"mat1/mat1_"+str(i)+".txt")
    time = f.read().splitlines()
    print(time)
    f.close()
    matData = pd.read_csv(str(dirc)+"mat1/mat1_"+str(i)+".csv")
    #for timestamp in time:
        #df.loc[(df.polarity >='1')&(df.polarity <='3'), 'num']= 'M'
    df = matData.loc[(matData['Timestamps'] >= time[2])&(matData['Timestamps'] <= time[3])]
    #print(df.iloc[len(df),1])
    print(df.shape)
    df
```

```
['1622711575.309220434', '1622711595.592306145', '1622711596.209033631', '1622711597.042408354', '1622711599.820118054', '1622711603.641375154', '1622711605.385108456']
(25, 2)
```

In [11]:
```
def substract(a, b):
    return "".join(a.rsplit(b))
```

In [9]:
```
dirc = 'A:/FYP/data/GelSight/GelSight/'
```

Out[9]:
```
'A:/FYP/data/GelSight/GelSight/'
```

In [12]:
```
substract(df.iloc[0,1],'/home/tas/Desktop/historical/processed/GelSight/')
```

Out[12]:
```
'mat1/mat1_50_frame_44.png'
```

In [189]:
```
from PIL import Image, ImageOps
import numpy as np
oneclass=[]
for i in range(1,51):
    f = open(str(dirc)+"mat12/mat12_"+str(i)+".txt")
    time = f.read().splitlines()
    #print(time)
    f.close()
    matData = pd.read_csv(str(dirc)+"mat12/mat12_"+str(i)+".csv",dtype="string")
    #for timestamp in time:
        #df.loc[(df.polarity >='1')&(df.polarity <='3'), 'num']= 'M'
    df = matData.loc[(matData['Timestamps'] >= time[2])&(matData['Timestamps'] <= time[4])]
    print(df.iloc[0,1])
    path = substract(df.iloc[0,1],'/home/tas/Desktop/historical/processed/GelSight/')
    #print(path)
    print(len(df))
    dirc = "A:/FYP/data/GelSight/GelSight/"
    savedirc="A:/FYP/data/GelSight/GelSight/CNN/"
    sample = []
    for ind in range(25):
        path = substract(df.iloc[ind,1],'/home/tas/Desktop/historical/processed/GelSight/')
        image = Image.open(dirc+path) # 用PIL中的Image.open打开图像
        zero = Image.open( "A:/FYP/data/GelSight/GelSight/mat1/mat1_1_frame_1.png")
        image_arr = np.array(image) - np.array(zero)
        #image = ImageOps.grayscale(image)
        #image_arr = np.array(image) # 转化成numpy数组
        image_arr = image_arr[100:340,150:470,:]
        sample.append(image_arr)
    sample_np = np.array(sample)
    oneclass.append(sample_np)
    oneclass_np = np.array(oneclass)
    print(oneclass_np.shape)
    with open('gelsight_mat12.npy', 'wb') as f:
        np.save(f, oneclass_np, allow_pickle=True, fix_imports=True)
```

In [30]:
```
from sklearn.model_selection import train_test_split
```

In [245]:
```
from einops import rearrange, repeat
from einops.layers.torch import Rearrange
trainset=[]
for i in range(4):
    num = i+1
    if num == 4:
        num = 12
    loaddata = np.load("gelsight_mat"+str(num)+".npy" )
    print(loaddata.shape)
    trainset.append(loaddata)
sample_np = np.array(trainset)
print(sample_np.shape)
trainset_np=rearrange(sample_np,'k n t h w c ->(k n) t h w c')
print(trainset_np.shape)
#with open("train_3class.npy", 'wb') as f:
 #   np.save(f, trainset_np, allow_pickle=True, fix_imports=True)
```

In [246]:
```
# get label
label=[]
for i in range(4):
    for k in range(50):
        label.append(i)
label_np = np.array(label)
print(label_np.shape)
print(label_np)
```

In [248]:
```
with open('25_60_80_3_train.npy', 'wb') as f:
        np.save(f, X_train, allow_pickle=True, fix_imports=True)
with open('25_60_80_3_trainlabel.npy', 'wb') as f:
        np.save(f, y_train, allow_pickle=True, fix_imports=True)
```

In [249]:
```
X_train, X_test, y_train, y_test = train_test_split(X_test, y_test, test_size=0.5, random_state=42)
```

In [250]:
```
with open('25_60_80_3_val.npy', 'wb') as f:
        np.save(f, X_train, allow_pickle=True, fix_imports=True)
with open('25_60_80_3_vallabel.npy', 'wb') as f:
        np.save(f, y_train, allow_pickle=True, fix_imports=True)
```

In [251]:
```
with open('25_60_80_3_test.npy', 'wb') as f:
        np.save(f, X_test, allow_pickle=True, fix_imports=True)
with open('25_60_80_3_testlabel.npy', 'wb') as f:
        np.save(f, y_test, allow_pickle=True, fix_imports=True)
```

In [244]:
```
from PIL import Image
import matplotlib.pyplot as plt
i=5
plt.figure()
#f, axarr = plt.subplots(3,1)
for k in range(3):
    plt.subplot(1, 3, k+1)
    image = loaddata[i][k]
    #plt.figure()
    plt.imshow(image)
    plt.show
    #img = Image.fromarray(image, 'RGB')
    #img.show()
```