CSC 486B Final Project Report – Team 9

Rylan Boothman, Avery Kushner, Oliver Tonnesen
University of Victoria
3800 Finnerty Rd, Victoria, BC V8P 5C2
{rylan, akushner, otonnesen}@uvic.ca

Abstract

We extend the Zero-shot face anti-spoofing model from Liu et al. *by applying their model to a zero-shot learning problem using the CIFAR-100 data set, by allowing their Deep Tree Network to have variable tree depth, and by removing the need for pixel-wise supervision masks in training data [3, 1]. We define zero-shot learning as a machine learning task whose goal is to classify examples from classes not seen during training. In testing, we found that the model struggles to achieve results significantly better than random guessing and never exhibits a loss curve that decreases uniformly throughout training. We hypothesise that the model's issues are caused either by the removal of pixel-wise supervision (meaning that the model can not generalize to data sets that do not contain maps for all training samples) or there is a bug in our code. Despite this, a significant drop off in accuracy between classes seen during training and classes unseen during training did not occur.*

1. Introduction

A face spoof attack is an attempt to deceive a facial recognition system by using a fake face to impersonate a genuine user or to hide an attacker's identity. Common spoof attacks include the use of makeup, masks, video replay, and printed photographs to fool recognition systems. Face anti-spoofing is designed to detect such spoofs before they're sent to a facial recognition system.

While defenses do exist for a wide range of known spoof attacks, they are usually trained only against one or two particular types of spoof attack, and are often ineffective when employed to detect any *unknown* spoof attacks. It is thus desirable to have as general a face anti-spoofing system as possible.

Liu *et al*. [3] aim to detect unknown spoof attacks using a Deep Tree Network (DTN) that partitions training samples into semantically similar subgroups and learns to classify each such subgroup individually.

This project aims to determine how general the proposed DTN is when applied to different zero-shot learning problems. To do this, the model was modified to work with a different set of data and for a different decision problem. Additionally, this project extends the network in a number of ways, including porting to a different machine learning framework and adding additional hyperparameters.

2. Contributions of the original paper

Where previous works used either handcrafted or CNN-based features to detect one or two types of face spoof attack, [3] extends current methods by removing the need for such supervised feature selection with their novel DTN, and collect a much more comprehensive database, comprising 13 distinct types of face spoof attack.

2.1. Deep Tree Network

The DTN proposed in [3] avoids the need to learn handcrafted or CNN-based features by first partitioning training data into semantic subgroups without supervision and based only on data variation. Once the partitions are made, the network learns to make its binary decision independently for each subgroup. Once a test sample is assigned to its most similar partition, the network uses that partition's learned binary classifier to determine whether or not the sample is a spoof.

The DTN is a full binary tree with height 3. Its nodes comprise three modules: the Convolution Residual Unit (CRU), the Tree Routing Unit (TRU), and the Supervised Feature Learning (SFL) module. Each non-terminal node in the tree consists of one CRU and one TRU; it routes each sample to either their left or right children based on data variation. Each terminal node in the tree consists of one CRU and one SFL module; it learns to predict whether or not a sample is a spoof using both the binary classification of spoof vs live, and also using a pixel-wise binary mask indicating the position of the spoof in the sample to help the network learn lower level features.

2.2. Zero Shot Face Anti-Spoofing with Multiple Spoof Types

Zero Shot Face Anti-Spoofing (ZSFA) attempts prior to [3] are trained using only 1 or 2 different types of spoof attacks. [3] expand on this by training their model using thirteen spoof types: replay, print, half-mask, silicone, transparent, paper craft, mannequin, obfuscation, impersonation, cosmetic, funny eye, paper glasses, and partial paper. Liu *et al*. hypothesis that the training with thirteen spoof attacks allows their model to be more robust against unknown spoof attacks.

2.3. Spoof in the Wild Database

Liu *et al*. also created the Spoof in the Wild database which represents a large variation in spoof attack types and provides a data set that future researchers my use as a bench mark. The data is split into two main super classes: *impersonation* attacks, whose goal is to have a sample be recognized as someone else, and *obfuscation* attacks, whose goal is to remove the attacker's identity from the sample. When collecting the samples for impersonation attacks, 720p video samples were taken from YouTube. The obfuscation attack samples are 1080p videos recorded directly by the authors. In total, the database consists of 1 630 five to seven second long videos.

3. Contributions of this project

This project extends the model detailed in [3] in a number of ways, including adding a hyperparameter to vary the depth of the DTN, removing the pixel-wise supervision in the SFL modules, and applying the model to a different zero-shot learning problem. Additionally, we port the model provided in [3] from TensorFlow2 to PyTorch [2] and refactor the training and testing pipeline to mirror the structure used throughout the assignments in CSC 486B/586B.

3.1. Variable Tree Depth

The DTN proposed in [3] has a hard-coded tree depth of 3, restricting the model to partition the training data into only eight subgroups. however, the optimal number of clusters may vary between data sets, so to allow users of the model to tune the depth of the tree to the optimal value, we made the tree depth configurable as a hyperparameter. To do this, each of the CRU, TRU, and SFL modules are initialized and accessed dynamically, and the amount of downsampling performed at each level of the tree is limited to ensure lower levels of the tree still receive trainable input.

3.2. Removal of Pixel-wise Supervision

The model in [3] uses a binary mask for each training sample that indicates where to look for the spoof. Examples of such masks can be seen in Figure 4 of [3]. The use of pixel-wise supervision masks during training likely helps the model to learn to identify spoofs, however it causes significant overhead to applying the model to other data sets as individual masks need to be created for every training sample. Additionally, the masks represent information that the model will not have at test time, and the use of such masks may not be possible when applied to certain data sets. For these reasons we chose to remove the pixel-wise supervision from the model. Thus, instead of taking an image, label, and binary mask as input during training, the model simply takes the image and corresponding label.

3.3. Application to Other Data

The original implementation of [3] on GitHub loads data from `.dat` files that are not present in the Spoof in the Wild Database. The Spoof in the Wild Database consists primarily of 15 second long QuickTime videos and does not have any `.dat` files in it, nor is the format of these `.dat` files anywhere documented. Since the model is designed for image classification and due of the time constraints of this project, we decided to apply the model to a new data set rather than convert the entire 200GB Spoof in the Wild database to a format usable by the model. Testing the model on a new data set has the added benefit of seeing if the model is applicable to other zero-shot learning problems.

Instead of the Spoof in the Wild Database we used the CIFAR-100 data set [1] that consists of 100 classes grouped into 20 super-classes. To simulate zero-shot learning we grouped the super-classes into two super-super-classes: natural and human-made objects and then dropped all training samples from 50% of the super-classes within each of our super-super-classes. At test time no samples were dropped. The natural super-classes are: aquatic mammals, fish, flowers, fruit and vegetables, insects, large carnivores, large natural outdoor scenes, large omnivores and herbivores, medium-sized mammals, non-insect invertebrates, people, reptiles, small mammals, and trees. The human-made super-classes are: food containers, household electrical devices, household furniture, large man-made outdoor things, vehicles 1, and vehicles 2. The choice of super-classes to drop during training is decided randomly when training begins. Because fourteen of CIFAR-100's super-classes are natural and only six are human-made, we balanced the training data by augmenting samples from the human-made classes using simple transforms such as vertical and horizontal flips.

4. Experimental Results

To tune the model we tested different numbers of filters, learning rates, weight initializations, Adam optimizer parameters, and tree depths. The final set of parameters was selected based on mean validation accuracy between the human-made and natural classes. We used the mean vali-

dation accuracy between the two classes rather than overall accuracy due to the severe class imbalance in the test data set. Because the CIFAR-100 data set does not have a predefined validation split, we used the first 1000 samples from the training data as the validation data set.

4.1. Model Training

The best mean validation accuracy was achieved with a learning rate of 10e-5, Kaiming normalization weight initialization, default Adam optimizer parameters, and a tree depth of three. The training and validation results are shown in Figures 1–4.

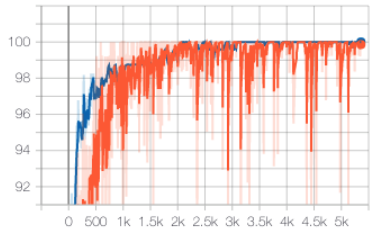

Figure 1. Human-made class training and validation accuracy

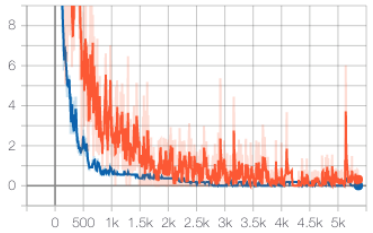

Figure 2. Natural class training and validation accuracy

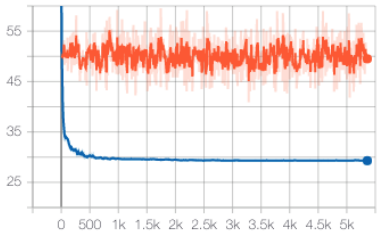

Figure 3. Overall training and validation accuracy

4.2. Test Results

The test results can be seen in Figure 5.The x-axis of Figure 5 shows the accuracy for just the Human-made class, just the natural class, and both classes together. The y-axis of Figure 5 shows the accuracy for just the sub-classes seen during training, just the sub-classes that were dropped during training, and all of the sub-classes together.

We hypothesised before training the model that higher accuracy would be achieved for the natural class than the human-made class due to class imbalance in the training data and attempted to avoid this by augmenting samples from the human-made class. However, as seen in Figures 1, 2, and 5 this clearly did not work.

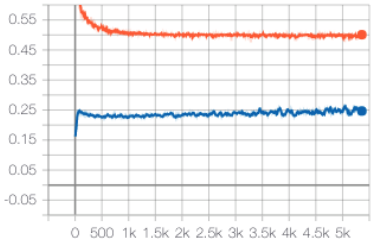

Figure 4. Overall training and validation loss

Despite this, the DTN achieves similar test accuracy between classes that were seen during training and classes that were dropped during training. A possible conclusion to draw from this is that the model succeeds at zero-shot learning, however because the test accuracy is only slightly better than what would be achieved by random guessing it could also be that the model has not learned anything and is just guessing randomly.

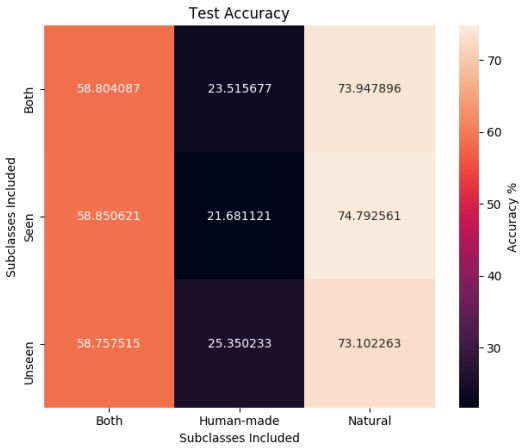

Figure 5. Test Accuracy

5. Future Directions

As can be seen in Figures 1 – 5 there are obvious issues in the application of the DTN to the CIFAR-100 data set and more work needs to be done to determine the source of these issues. We hypothesise that the cause of the issues is either the removal of the pixel-wise supervision masks, which would mean that the DTN cannot be applied to a data set that does not have these. To test this hypothesis we would need to create the pixel maps for the CIFAR-100 data set and attempt to re-train the model using them. Due to the time constraints of this project, this was not possi-

ble. Our other hypothesis to explain why the DTN is not working is that we introduced a bug either while porting the model PyTorch or while modifying the tree depth parameter and allowing it to vary. However, we were not able to find any obvious bugs in the time we had.

5.1. Zero-shot learning for Multi-class Classification

Since the CIFAR-100 data set is already grouped into classes and super-classes, zero-shot learning could be applied directly to it without our super-super-classes by dropping all the samples from half of the classes within each super-class at training time and then making a multi-class classification of each sample among the twenty super-classes. However, the deep tree network is inherently designed for binary classification and the amount of modification required for the tree routing algorithm to handle multiple-classes was beyond the scope of this project.

References

[1] A. Krizhevsky. Learning multiple layers of features from tiny images. Technical report, 2009.

[2] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. dAlché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[3] A. J. X. L. Yaojie Liu, Joel Stehouwer. Deep tree learning for zero-shot face anti-spoofing. In *In Proceeding of IEEE Computer Vision and Pattern Recognition (CVPR 2019)*, Long Beach, CA, 2019.