ClassroomNet: Learning Generalizable, Interpretable Features via Stacked Knowledge Distillation

George Tang
gtangg12@mit.edu

Michael Zhang
msz@mit.edu

William Zhao
wzhao6@mit.edu

Abstract

Ensuring that features learned by deep neural networks specialized to one task are generalizable, robust, and interpretable is ill-posed. This is due to the complexity of the parameter space, lack of ability of the network to differentiate between high and low-quality feature sets given the same performance, as well as lack of task variety and therefore learned feature variety. Recent advancements have shown that distilling knowledge from a variety of tasks on consistent representations of input via teacher-student models can greatly increase performance. Inspired by this idea, we propose ClassroomNet, which seeks to distill higher-quality features from pretrained teachers into a student model that is responsible for the downstream task. Furthermore, the student learns which teachers are useful, allowing us to formulate ClassroomNet as a generalizable, interpretable paradigm. However, we were unable to successfully train ClassroomNet empirically on the Waymo Perception dataset (please see Section: Other Notes). All our code are available at `https://github.com/gtangg12/classroom-net`.

1. Introduction

The classical deep learning paradigm is to learn representative features directly from one task or a small set of related tasks, as in the case of transfer learning or multi-task learning. However, the lack of task diversity usually results in features that fail generalize to other tasks [10], are less robust, and are less interpretable by humans. Moreover, even if a network has ability to represent higher-quality features, they are often extremely difficult to learn.

Several solutions have been explored. One option is to pass human-interpretable features such as depth and segmentation maps as auxiliary input [2] to the network. Intuitively, human-interpretable features may perform better compared to purely learned features since humans generalize much better and are more robust than state-of-the-art computer vision models. Modern approaches employ deeper models, since they have higher learning capability and implicit regularization [7]. However, they may still have difficultly learning features outside of those immediately determined by the task formulation, and interpretability still remains an issue.

There has also been significant progress in knowledge distillation, during which the teacher network with higher-quality features guides the student to learn a consistent mapping between the student's input and teacher's features [1], which might be ill-posed to learn alone. The student can then combine its learned teacher features as well as its own features learned under the constraint of mimicking teacher features for downstream tasks [6].

Recently for vision tasks, it has been shown that distilling from networks specialized to other tasks on consistent representations of the input (e.g. RGB image, correspoding LIDAR point cloud projection) can greatly improve performance [6], [10]. The intuition is that the intermediate representations used to solve the other tasks may also be useful for the current task even though there is no immediately obvious way to learn them using the current task formulation and network architecture.

2. Motivation

We propose ClassroomNet, which expands upon the architecture in [6], in an attempt to address the issues highlighted in the introduction. In particular, we hypothesize that pretrained teacher models that specialize on a diverse set of tasks provide robust features in aggregate, and that these latent representations can be distilled into a student model in order to improve performance. In addition, the existence of multiple teacher networks also allows the student to selectively learn the teacher representations that are most conducive to the downstream task. Each teacher can be viewed as either enforcing human interpretable features (e.g. ensuring the student is able to reproduce the segmentation map), increasing feature variety (e.g. 3D to 2D distillation [6]), or helping guide the student to learn its own higher-quality features that are fused with teacher features.

3. Architecture

The architecture of ClassroomNet is shown in Figure 1. The design is based on `torchvision`'s ResNet18 and Faster R-CNN implementations as well as the distillation architecture provided by [6]. We made several key modifications to extend their components to our situation, which include:

- increasing flexibility in the student module to support multiple teacher features,
- connecting the student module to the Faster R-CNN architecture for sophisticated aggregation by treating the student module as the Faster R-CNN's backbone model, and
- adding the depth prediction head to Faster R-CNN, which is based on a loss function that more heavily weighs smaller depths, since otherwise their contribution will be overshadowed by objects at farther distances.

3.1. Student Backbone

First, we utilize a few ResNet [4] layers in the student module; we chose to utilize part of the ResNet architecture due to its verified ability to perform well on image recognition tasks. The input is then processed by three channels in parallel in order to learn features corresponding to the two pretrained teacher models and other miscellaneous features. For the two learned feature modules in the student network, we also employ normalization modules based on the pretrained feature values in order to aid the student model in reproducing these features. We utilize L2-loss for the distillation modules, as was done in previous 3D-to-2D distillation work [6].

We have removed all pooling and downsampling layers from the ResNet layers in the student module in order to learn features of dimension $C \times H \times W$, where C is the number of channels within the learned feature and H and W are the original image's height and width, respectively. This is done so that the dimensions of the learned features match up with the dimensions of the pretrained features, allowing for simpler knowledge distillation. Furthermore, due to computational limitations, we truncated the ResNet to only a single convolutional layer followed by two ResNet blocks.

3.2. Aggregation and Prediction

To re-aggregate the three sets of features and produce bounding box predictions, as well as object classifications and depth estimates for each bounding box, we choose to use a Faster R-CNN [9] architecture since it is specialized for the first two of theses tasks. Specifically, we utilize the feature pyramid network (FPN), region proposal network (RPN), and Fast R-CNN components of the Faster R-CNN architecture while exchanging the simple ResNet backbone for our student module. We also augment the existing Faster R-CNN implementation with an extra output head in order to generate depth predictions as well.

In addition, the outputs for the bounding box, class, and depth predictions have shapes $N \times M \times 4$, $N \times M$, and $N \times M$, respectively, where N is the number of object detections and M is the number of object classes. In particular, this means that one bounding box and one depth is associated with each object class for each detection.

To facilitate better training, we predict a normalized depth instead of depth using the student model. In particular, we used the function $(\frac{d}{d_{\max}})^{\frac{1}{2.2}}$ to normalize the depth regression target to be between 0 and 1, with a stronger focus on smaller depths. We had also considered other methods of normalization, such as taking the logarithm $\log(d)$, but eventually decided against this since we'd like to keep the depth loss bounded at the size of around 0.1 as we'd like to add together and simultaneously optimize many losses (such as our distillation loss), and would like all our losses to be equally optimized. We also clamp the depth between 0 and 1 to remove any outliers with infinite depth values.

Lastly, following Faster R-CNN [9], we use the smoothened L1-loss on the depth regression task to prevent outliers in depth from having a strong effect on the predictions.

3.3. Teacher Models

3D SPVNAS Model We use SPVNAS [11] as our 3D teacher model. We chose this model due to its state-of-the-art performance and fast inference speed. We have decided against our original choice of PointPillars [5] since we found that the pillars were not well aligned with the 2D image inputs of the student models, and that its underlying Poin tNet backbone [8] has less capabilities than SPVNAS to model 3D objects due to the lack of convolutional layers.

SPVNAS generates features for each voxel in 3D space which is inhabited by at least one point in the point cloud. We convert this into features in 2D space by assigning to each point in the point cloud the features from its corresponding voxel, then projecting each point in the cloud to the 2D image. Since the camera and the LIDAR overlap, each pixel in the 2D image bijectively correspond to a point in the LIDAR point cloud, allowing us to do this final projection.

Mask R-CNN Model We use Mask R-CNN [3] as our image segmentation 3D models. Due to disk space and compute limitations, we were not able to store precomputed Mask R-CNN features nor compute them live. Therefore, we opt instead to use the masks generated by the pretrained model. We believe that these masks are easier to generate

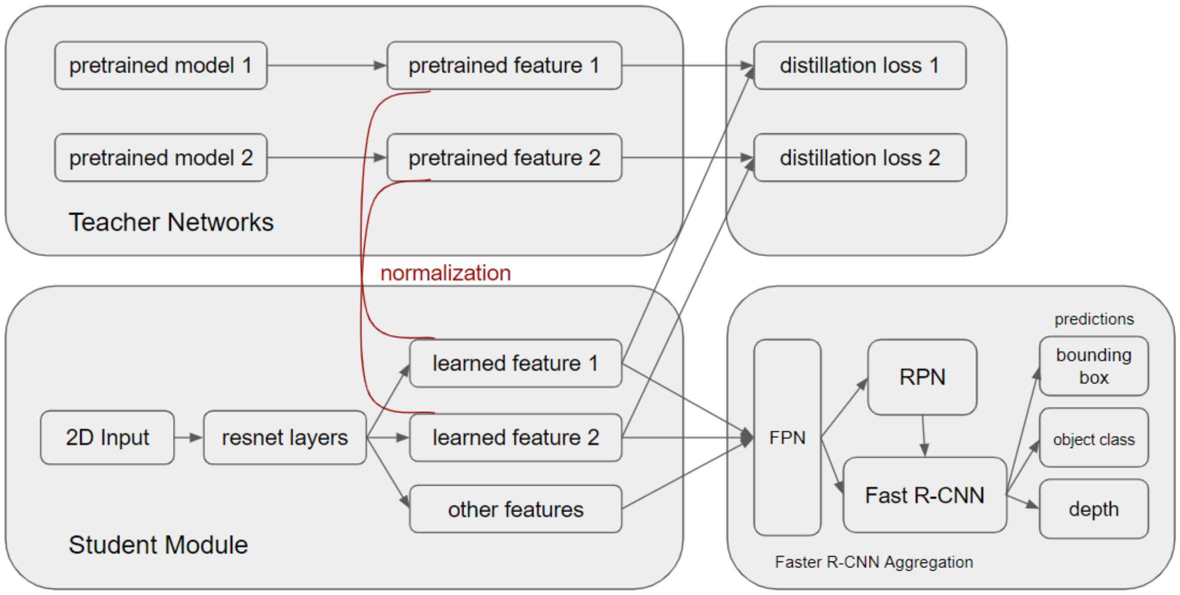

Figure 1. ClassroomNet architecture schematic.

than depth predictions, but also provides useful information to refine the model's depth estimates since it provides information on the sizes and types of objects in the image. In particular, we save the segmentation masks for the 5 most common classes given by the pretrained model.

Mask R-CNN returns a list of bounding boxes along with the segmentation mask and class associated to each bounding box. In order to obtain global segmentation masks, we use the bounding box prediction confidences to pick the most likely masks. We start by labelling all pixels as background pixels. Then, iterating through all bounding boxes from lowest to highest probability, we set the most likely segmentation class for any pixel in the segmentation mask for that bounding box to be the class for the bounding box.

4. Experiments

We now describe our experimental design and implementation. Due to severe constraints (see Section: Results and Discussion and Section: Other Notes), we were unable to rigorously evaluate our framework. We describe the experiments we would've run and the reasoning behind them as well as note which ones we were unable to get to.

4.1. Dataset and Implementation

We use the Waymo Self-Driving Perception Dataset for our experiments due to its scale, diversity, and variety of features. Our inputs and teacher features are derived from the front camera RGB image and the corresponding points in the sparse point cloud that projects onto the front image. Specifically, the front camera image is the input to the student model and input to the segmentation teacher model, and the point cloud is used as input to the 3D teacher model. Our labels include object classes and their respective the bounding boxes. In addition, to generate the depth values for detected objects, we simply average depths of the respective points whose projection lies in the bounding box.

In total, we involved around $50,000$ datapoints, split into $48000/2000$ for the train and validation distributions. We did not think that a testing distribution makes sense for us since we train on only 1 epoch, so model has no chance to overfit to either of training or validation sets. Preprocessing of the point clouds was done on the MIT Satori cluster with 128 CPUs. Since the mask features generated by the segmentation teacher model don't take much disk space, we precompute them to avoid running Mask R-CNN, which runs only a few frames per second, repeatedly. The 3D model runs very efficiently on GPU, but its features take much disk space, so we choose to repeatedly run it instead of precomputing teacher features.

4.2. Interpretability

Ablation To evaluate the efficiency of our model, we compared it with another model which is trained using the same student architecture and hyperparameters, but without any information from the teacher models. Due to our limited computation capacity, we were only able to compare the performances of our model and the student model after 1 epoch of training. We show the plots of each model's training loss over time for every batch. Note that there was

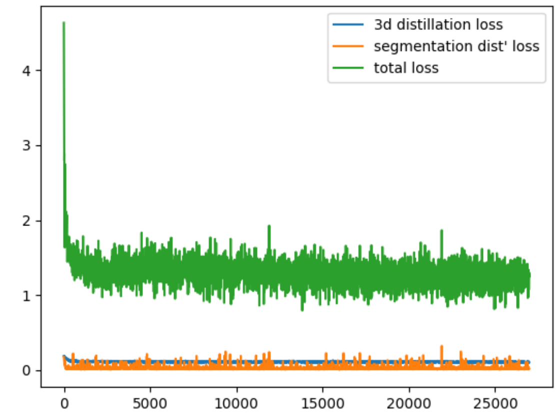

Figure 2. Train Loss of Normal Model vs Number of Datapoints Seen

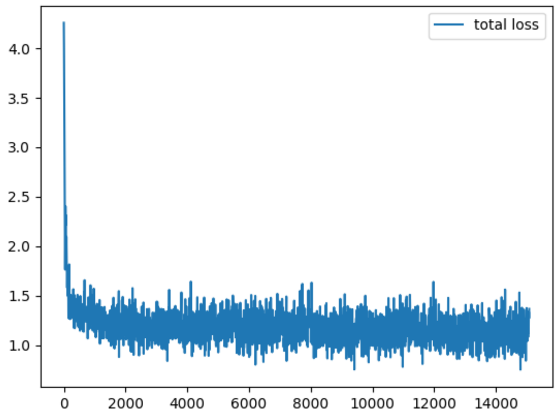

Figure 3. Train Loss of Ablated Model vs Number of Datapoints Seen

no need to use a separate testing dataset here since each batch in the training set is unseen during the first epoch. We show the plots in figures 2 and 3.

Fine-grained Analysis In addition to ablation, we can perform fine-grained analysis on which features from which teachers are useful to the current task at hand. Specifically we can analyze the learned kernels of the first convolution layer of the region proposal network. Applying them to the learned feature channels, we can see which channels induces a response for which masks and draw conclusions accordingly. However, we were unable to complete this experiment due to the lack of a good trained model.

4.3. Generalizability

Since our models create features that fit human intuitions, we would expect our model to have learned more generalizable features. We intended to use linear-probing/fine-tuning to determine the generalizability of our features. Again, we were unable to complete this experiment due to the lack of a good trained model using our architecture.

4.4. Robustness

We also investigate whether our model architecture is more robust to adversarial attacks. We hypothesize that because it uses more human understandable features, such as segmentation masks, it would be more robust to adversarial attacks. We wanted to test our model adversarially using the Iterative Gradient Descent method. To understand the effects of distillation losses on adversarial attacks, we intended to separately attack each component of our loss function to find the comparative robustness between the different losses in our model. However, as with generalization experiments, this could not be done due to the lack of a well trained model.

5. Discussion

As shown in Figure 2, the training distillation loss does not improve over time. This can explain why there appears to be little difference in the performance of the two models – if ClassroomNet cannot successfully learn the teacher features, then having those extra learned feature components provides no advantage over the ablated model. This observation is also substantiated by the values shown in Table 1.

Perhaps a more viable approach is to repeat the framework for a simpler dataset like sign-language recognition, which removes the complexity of self-driving in terms of diversity of data as well as compute since the images are simpler, smaller, and there are less confounding variables, so within the first few epochs, the model will converge, unlike our current task, which most likely requires thousands of epochs with extremely low learning rate to converge. The point clouds can be easily collected via a Microsoft Kinect and the segmentation will only be a binary mask: hand vs background.

6. Team Contributions

George worked on the infrastructure of the project. He developed the datalake, which includes processing on Satori and storing the waymo frames into images, point clouds, labels, and depths. He also wrote the dataset infrastructure, which provides and interface with the datalake for tasks downstream. He also worked on getting the pipelines up and running on the compute platforms. Finally, he helped with the design of the overall architecture of ClassroomNet.

Model	Class	Box Reg	Depth	Obj	RPN Box Reg	3d Dist	Mask Dist
Normal	0.188	0.219	0.082	0.218	0.428	0.107	0.026
Ablated	0.183	0.186	0.083	0.246	0.456	NA	NA

Table 1. Loss values for ClassroomNet vs ablated ClassroomNet (no teachers) after 1 epoch

William worked on the pretrained model zoo, including the both teacher models SPVNAS and Mask R-CNN, as well as the preprocessing and postprocessing code to generate features which match the input shape, such as the projection from point cloud and the generation of global segmentation masks. This involves understanding and choosing appropriate teacher models for each teacher task and understanding their input and feature formats. William also created all plots and tables shown in this report.

William and Michael co-wrote the code for training. William and Michael also worked together to fit together all the code written by the 3 members and run them on AWS compute platforms.

Michael completed the implementation of the ClassroomNet architecture, which involved finding and parsing code from existing implementations of ResNet, Faster R-CNN, region proposal networks, and feature pyramid networks in order to make the necessary changes and combine them into the final ClassroomNet.

All contributed to writing the paper.

7. Other Notes

Despite having the paper and code[1] ready, we were unable to train and evaluate our model rigorously. This was mainly due to time and compute limits. First, we were working with self-driving, so the dataset is necessarily huge and diverse (e.g. road conditions, time-of-day, weather, etc.). Despite managing to preprocess all the features and building a datalake, our original compute platform, Google Colab Pro with 1TB Google Drive storage, could simply not meet our specifications in terms of file transfer speed, file loading speed, and reliability.

We also tried running on MIT Satori cluster. The preprocessing worked fine. For the training, we managed to get everything working with the exception of `torchsparse`, a package that is needed to run our 3D point cloud teacher model. We resorted to manually building the package and its dependencies locally, since we do not have sufficient permissions and conda lacked the packages for the Linux distribution used by Satori. However, at the end, there was an unresolvable issue with `usr/lib/bins/floatn.h` missing some datatype, causing nvcc to fail to compile the the cuda version of the library.

Finally, we resorted to AWS, but the model required a long time to train on our limited AWS resources; in particular, we believed that it could take hundreds of epochs for losses to converge, but our AWS cluster could only process one epoch per hour. The model ultimately required significant hyperparameter tuning as well, which we were unable to accomplish. Specifically, every part of the module seems to be working except for the region proposal network, which may be due to some issue with normalization or that is simply requires more data/training time. We also had to scale down our model significantly in order to not run out of memory on the AWS GPU, but we do not think that this is the main reason for failure in training. Furthermore, there was an widespread AWS outage on Tuesday (check MIT AWS outage email that said attestation systems are down), that blocked further experimentation.

[1]Code available at `https://github.com/gtangg12/classroom-net`.

References

[1] Sajjad Abbasi, Mohsen Hajabdollahi, Nader Karimi, and Shadrokh Samavi. Modeling teacher-student techniques in deep neural networks for knowledge distillation, 2019. 1

[2] Vitor Guizilini, Rares Ambrus, Sudeep Pillai, Allan Raventos, and Adrien Gaidon. 3d packing for self-supervised monocular depth estimation. In *CVPR*, 2020. 1

[3] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn, 2018. 2

[4] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015. 2

[5] Alex H. Lang, Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang, and Oscar Beijbom. Pointpillars: Fast encoders for object detection from point clouds, 2019. 2

[6] Zhengzhe Liu, Xiaojuan Qi, and Chi-Wing Fu. 3d-to-2d distillation for indoor scene parsing, 2021. 1, 2

[7] Behnam Neyshabur. Implicit regularization in deep learning, 2017. 1

[8] Charles R. Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation, 2017. 2

[9] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks, 2016. 2

[10] Jing Shao, Siyu Chen, Yangguang Li, Kun Wang, Zhenfei Yin, Yinan He, Jianing Teng, Qinghong Sun, Mengya Gao, Jihao Liu, Gengshi Huang, Guanglu Song, Yichao Wu, Yuming Huang, Fenggang Liu, Huan Peng, Shuo Qin, Chengyu Wang, Yujie Wang, Conghui He, Ding Liang, Yu Liu, Fengwei Yu, Junjie Yan, Dahua Lin, Xiaogang Wang, and Yu Qiao. Intern: A new learning paradigm towards general vision, 2021. 1

[11] Haotian Tang, Zhijian Liu, Shengyu Zhao, Yujun Lin, Ji Lin, Hanrui Wang, and Song Han. Searching efficient 3d architectures with sparse point-voxel convolution. *CoRR*, abs/2007.16100, 2020. 2