Object Classification for Autonomous Vehicles based on Learning by Ignoring

Joseph Chang, Benjamin Chang
Department of Electrical and Computer Engineering, University of California, San Diego
jdchang@ucsd.edu, bmc011@ucsd.edu

Abstract

Machine learning models aim to emulate the the way people learn to perform meaningful tasks. One such task is the ability to ignore less relevant information when trying to recognize things in the world. This paper investigates a machine learning algorithm that can imitate this aspect of human recognition. We aim to apply this model to recognize street objects such as cars, people, buses, and bicycles to increase the safety of autonomous vehicles. We demonstrate that it results in improved performance compared to models which do not ignore less relevant information while learning.

1 Introduction

Machine learning is a rapidly growing field due to the numerous important problems it is able to solve. Due to improved hardware and huge increases in data over recent years, deep learning has risen in popularity. Computer Vision is no stranger to this and has adopted many machine learning algorithms to help computers analyze and understand the environment through images and video. One problem that has largely been solved is the ability for computers to recognize vehicles or pedestrians from video footage or images. This is useful for autonomous vehicle applications as it allows a car to know to brake during a life-threatening situation. Although there has been much progress in these fields, there are still many challenges to the accuracy of recognition algorithms such as poor lighting, occlusion, differing viewpoints, and background clutter. In most of these scenarios, there is some data which is nearly irrelevant to recognizing a vehicle or pedestrian, yet the network trains on it. This decreases performance and can slow down a model's ability to detect an object quick enough for real-time applications. Thus, we learn to ignore data that is irrelevant to a recognition task and demonstrate it classifies roadway obstacles more accurately. This will increase the safety of people and goods on the road and is an important problem to solve as autonomous vehicles will eventually be used in life-threatening situations.

2 Related Works

Interleaving Learning. There has been some recent investigation into machine learning models which are trained to imitate some aspect of human learning. One example is interleaving learning [1] which proposes a method for domain adaptation. It uses the fact that humans apply what they know about a certain object to recognize similar objects in a different setting or domain. This increases long-term retention and improves ability to transfer learned knowledge. In interleaving learning, multiple models learn a data encoder collaboratively by interleaving encoders to pass newly learned weights through previous encoder steps. For instance, when there are three models, the encoder is trained by model 1 for a while, then passed to model 2, and finally to model 3. This process is repeated for multiple rounds creating a multi-level optimization network that achieves state-of-the-art performance on several benchmark datasets.

Learning-By-Passing Tests. Another paper was inspired by the way humans learn by passing tests [2]. Taking exams is a globally known method to effectively improve people's learning outcome. A sequence of tests with improving difficulty gives people a benchmark of where they are at and helps them identify weak areas that need work so they can continue to improve by targeting those weak points. In essence, a tester model is created to make increasingly difficult tests to evaluate a model as it is being learned. The learner model tries to continually improve to pass each test successfully and is a multi-level optimization framework like interleaving learning [1].

Learning-By-Self-Explanation. Learning by Self-Explanation (LeaSE) [3] is inspired by the way students deepen their understanding of a topic by explaining it to themselves. This is a powerful technique used to increase memory retention. LeaSE uses an explainer model to try to clearly explain to an audience model how to get to a prediction outcome. It is formulated as a multi-level optimization framework which involves several stages of learning. First, the explainer learns how to get to a particular outcome. Then, it explains that process to the audience model being learned. The explainer and audience then validate that they are understanding each other. This algorithm can be used for image classification problems to achieve state-of-the-art performance.

Small Group learning. Small Group learning [4] leverages the human learning skill where people work in small groups together to improve their learning capability. When there's more people, each person can express their specific perspective on a topic, compare their ideas with peers, or solve problems together. It is composed of multiple learners. Each uses its intermediately trained model to generate a pseudo-labeled dataset, then retrains its model using pseudo-labeled datasets generated by other learners. It is formulated as a multi-level optimization framework with three learning stages. First, the learners train their network weights independently. Then, they train network weights together with pseudo-labeling. Finally, they improve their individual architectures by minimizing the validation losses. This algorithm was demonstrated to be effective on the CIFAR-100, CIFAR-10, and ImageNet datasets.

Learning-By-Ignoring. Recently, there have also been papers on learning by ignoring [5] to create a model that imitates how humans automatically ignore less relevant data to recognize activities and estimate pose more accurately. This trains two models using a 3-level optimization framework with pretraining, finetuning, and updating steps. It begins by identifying data that has a large domain shift from the target distribution and learns to ignore them with an ignoring variable. Then, it trains a model on finetuning data with learning by ignoring and another on target data with learning by ignoring. Finally, an L2 norm is used to make the weights of the second model closer to the first.

3 Method

To evaluate the validity of object recognition using Learning-by-Ignoring (LBI) [5], we compare the performance of two trained models.

For data, we use the *Visda2017* dataset [6] which contains image data of 12 different objects or classes including cars, buses, and people. The dataset consists of 2 domains, training and validation. The training images are CAD models of objects against a white background. The validation images are regular RGB photos of the street objects. This is shown in Fig. 1.

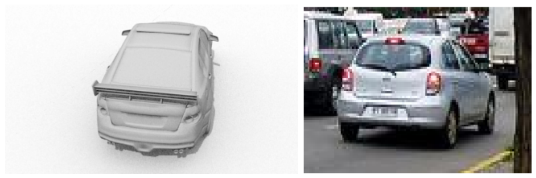


Figure 1: Visda2017 Training and Validation Data

To train recognition models for specific objects, we use a three-level optimization framework which contains three training steps. First, pretraining minimizes a loss function weighed by an ignoring variable. The ignoring variable remembers which types of pixels are less effective for classifying an image. By excluding less relevant data, the pretraining model is finetuned since not all data is good for training. Not all data is useful because there is typically a difference between pretraining and finetuning domains.

An example of domain difference is seen by comparing our training images with our validation images where training has white background and validation has colored background. Accounting for this difference while training our model leverages the different distributions of each domain and further improves the model of the training dataset to classify images from the validation domain. This causes the model to learn to ignore irrelevant data by giving a weight to the pretraining dataset's loss.

The model V is trained with LBI on the training domain and the model W is trained normally on the validation domain. Then, we use the L2 norm $||W - V^*(A)||_2^2$ where A is the ignoring variables $A = \{a_i\}_{i=1}^M$. This lets the weight of W be close to V's weight with the goal of having W perform well on the validation dataset.

The reason we want to train V with LBI is that we want to learn the weights that are useful for W. Doing these steps trains a final LBI finetuning model which is used for evaluation on test data by making predictions on vehicle scenario classes and updating the the ignoring variable by minimizing the validation loss. This model is then a useful representation of finetuning image information that is relevant to the finetuning dataset.

Of the three models, Model 1 ignores the pretraining dataset and directly trains a model on the finetuning dataset. Model 2 combines pretraining and finetuning datasets and trains a model on this data. It first trains Model M1 on the pretraining dataset and M2 on the finetuning dataset. Then, M2 is encouraged to be close to M1 by minimizing their squared L2 distance to get Model 2. Model 3 is the LBI model described above which will be compared to the first two baseline models.

The general pipeline for training the LBI model is shown in Fig. 2 where we see the pretraining, finetuning, and validation steps. Solid arrows represent making predictions and calcualating losses. Dotted arrows show updating parameters by minimizing losses. The pipeline equations for the encoder, ignoring variables, and model training are shown below.

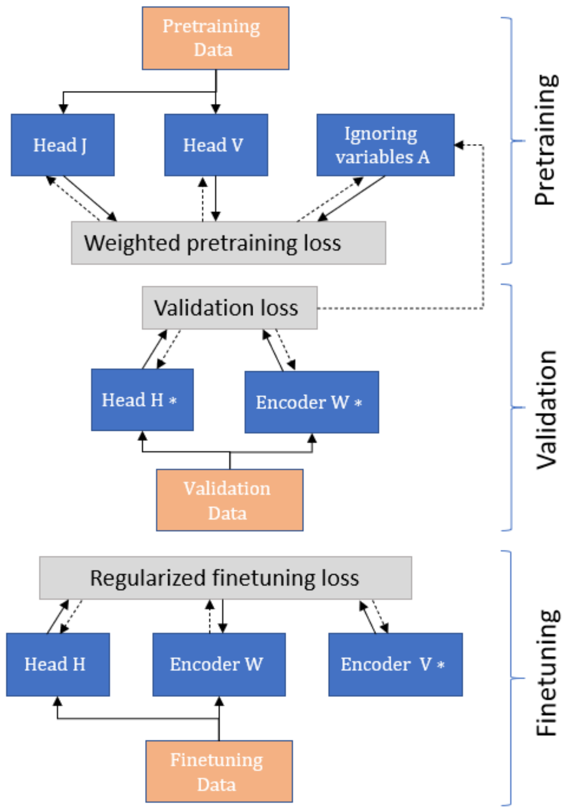


Figure 2: Learning-By-Ignoring Pipeline

We train models with three training stages. First, we pretrain a data encoder V as follows where $V^*(A)$ is the optimal solution, A is the ignoring variable, and V is the data encoder. The loss of each pretraining example is multiplied by its ignoring variable a. The ignoring variables $A = \{a_i\}_{i=1}^M$ are fixed at this stage, but updated later.

$$V^*(A) = \min_{V,J} \sum_{i=1}^{M} a_i L(V, J, d_i^{(\text{pre})})$$

In the second stage, we finetune because not all pretraining examples $d_i^{(pre)}$ are good for training W due to domain differences. We associate each pretraining example with an ignoring variable b=[0,1] to determine if it should be ignored while training W with the equations below. Then, we train the data encoder W on a finetuning dataset $D^{(tr)}$. This encourages the weight of W to be close to the optimal encoder $V^*(A)$ by minimizing the squared L2 distance $||W - V^*(A)||_2^2$ between W and $V^*(A)$ where A are the ignoring variables. This helps W perform well on validation.

$$\begin{aligned}
\min_{A,B} \quad & \textstyle\sum_{i=1}^{O} L(W^*(V^*(A), B), H^*(B), d_i^{(\text{val})}) \\
s.t. \quad & W^*(V^*(A), B), H^*(B) = \min_{W,H} \sum_{i=1}^{N} L(W, H, d_i^{(\text{tr})}) + \lambda \|W - V^*(A)\|_2^2 + \gamma \sum_{i=1}^{M} b_i L(W, H, d_i^{(\text{pre})}) \\
& V^*(A) = \min_{V,J} \textstyle\sum_{i=1}^{M} a_i L(V, J, d_i^{(\text{pre})})
\end{aligned}$$

We optimize W using the following where λ is a tradeoff parameter.

$$W^*(V^*(A)), H^* = \min_{W,H} \textstyle\sum_{i=1}^{N} L(W, H, d_i^{(\text{tr})}) + \lambda \|W - V^*(A)\|_2^2$$

For the third stage, we apply $W^*(V^*(A))$ to make predictions on the validation dataset $D^{(val)}$ of a target class. Then, we update A by minimizing the validation loss.

$$\min_A \sum_{i=1}^{O} L(W^*(V^*(A)), H^*, d_i^{(\text{val})})$$

Combining all three stages together, we get the following equation which nests the three optimization problems together. This problem is solved with a combination of gradient descent steps and approximations. The resulting optimization algorithm pseudocode is shown in Fig 3.

$$\begin{aligned}
\min_A \quad & \textstyle\sum_{i=1}^{O} L(W^*(V^*(A)), H^*, d_i^{(\text{val})}) \\
s.t. \quad & W^*(V^*(A)), H^* = \min_{W,H} \textstyle\sum_{i=1}^{N} L(W, H, d_i^{(\text{tr})}) + \lambda \|W - V^*(A)\|_2^2 \\
& V^*(A) = \min_{V,J} \textstyle\sum_{i=1}^{M} a_i L(V, J, d_i^{(\text{pre})})
\end{aligned}$$

Algorithm 1 Optimization algorithm for learning by ignoring

```
while not converged do
    1. Update encoder $V$ in the pretraining phrase using Eq.(6)
    2. Update task-specific head $J$ in the pretraining phrase using Eq.(7)
    3. Update encoder $W$ in the finetuning phrase using Eq.(8)
    4. Update task-specific head $H$ in the finetuning phrase using Eq.(9)
    5. Update ignoring variables $A$ using Eq.(10)
end
```

Figure 3: LBI Training Algorithm

$$V' = V - \xi_V \nabla_V \sum_{i=1}^{M} a_i L(V, J, d_i^{(\mathrm{pre})}) \tag{6}$$

$$J \leftarrow J - \xi_J \sum_{i=1}^{M} a_i \nabla_J L(V, J, d_i^{(\mathrm{pre})}) \tag{7}$$

$$\begin{aligned} W' &= W - \xi_W \nabla_W (\textstyle\sum_{i=1}^{N} L(W, H, d_i^{(\mathrm{tr})}) + \lambda \|W - V'\|_2^2) \\ &= W - \xi_W (\textstyle\sum_{i=1}^{N} \nabla_W L(W, H, d_i^{(\mathrm{tr})}) + 2\lambda(W - V')) \end{aligned} \tag{8}$$

$$\begin{aligned} H' &= H - \xi_H \nabla_H (\textstyle\sum_{i=1}^{N} L(W, H, d_i^{(\mathrm{tr})}) + \lambda \|W - V'\|_2^2) \\ &= H - \xi_H \textstyle\sum_{i=1}^{N} \nabla_H L(W, H, d_i^{(\mathrm{tr})}) \end{aligned} \tag{9}$$

$$\begin{aligned} A &\leftarrow A - \xi_A \nabla_A \textstyle\sum_{i=1}^{O} L(W', H', d_i^{(\mathrm{val})}) \\ &= A - \xi_A \textstyle\sum_{i=1}^{O} \frac{\partial V'}{\partial A} \frac{\partial W'}{\partial V'} \nabla_{W'} L(W', H', d_i^{(\mathrm{val})}) \end{aligned} \tag{10}$$

4 Technical Implementation

We implement the project by first running the original Learning-By-Ignoring code to ensure it runs properly and achieves correct results. This involves downloading required packages, creating the code environment, and downloading the Office31/Officehome datasets. The original datasets were not split into training, testing, and validation sets so we created a split file which could do that. Specifically, we split training, validation, and testing with a 6:2:2 ratio respectively. Since each class of the Visda2017 dataset has far more images than the Office31 dataset, we only used every other 38 images in Visda2017 for training. The ratio of Visda2017 images to Office31 after selection is 38. This prevents a very long training time. Due to the nature of the data splitting program, it only ran in a local environment so we moved the split data to our cloud-based environment later. After splitting the data, we also generated .txt files with the paths to our image files for the LBI program to know where to access images. In doing this, we needed to account for the need to format paths correctly in Linux vs Windows.

We cloned the original LBI repository, debugged environment issues, and were able to attain the results of the Learning-By-Ignoring paper, though with slightly less accuracy. This was because our computational memory maxes out at 32 image batches while the original results were run with 64 image batches. We select the Visda2017 dataset as our custom dataset to create a new Learning-By-Ignoring prediction model. The data includes a variety of vehicles such as bicycles, buses, cars, and people. To preprocess the data, we use the same 6:2:2 split as with the office31 data and modify the original main Learning-By-Ignoring python file to be compatible with the new dataset. This includes creating new domain dictionaries, creating new bash files to train our models iteratively, adding code to core files like Visda2017.py, etc. The domain dictionaries for the Visda2017 dataset are T for the training domain and V for the validation domain. We train each model for 50 epochs and compare the results with that of the original Learning-By-Ignoring paper. The model we train with is a ResNet [7] model which is shown in Fig. 4.

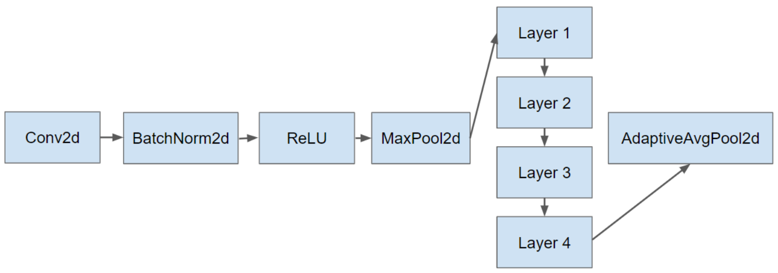


Figure 4: Visda2017 Model Results

In the above figure, Conv2d is a 2D convolutional layer which creates a 2D convolution that is convolved over input images. BatchNorm2d is a layer which makes the RNN faster by improving the learning rate and more stable by recentering and rescaling each layer's distribution as the parameters of the layer above it change during training. The ReLU layer acts as an activation function for the network by determining if nodes in it should be activated or not based on inputs received. MaxPool2d is a layer which downsamples the input to reduce dimensionality and allows for more variation in the output model as the layers become deeper. It does this by taking the maximum value over the window defined by the pool size for each dimension along the features axis. Layers 1-4 are contain many 3D blocks of 3x3 convolution layers of varying dimensions which store features from input. Lastly, AdaptiveAvgPool2d is an Adaptive Average Pooling Layer which acts similar to a MaxPool layer by downsampling inputs, but it is specified by an output size while its kernel size and stride size are left to be automatically adapted to the network's needs.

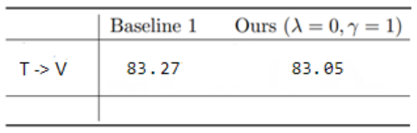
5 Novelty

Our research is novel since it applies the Learning-by-Ignoring algorithm [5] to a new type of data, vehicles and street objects. This involves writing custom code and modifying core LBI code to train models specific to vehicle scenarios. It expands the application of LBI which was previously limited to human activity recognition and pose estimation. However, we show LBI can be used to improve machine learning algorithms in vehicle scenarios with good accuracy. It is a robust algorithm for use in a broader range of real-world scenarios.

6 Results

To test LBI, we train 2 custom models on the Visda2017 vehicle dataset. The first model uses the "Ours2" training type which trains a LBI classification model using specified pretraining data. The second model uses the "Baseline 1" training type which trains a classification model with specified finetuning data. For both models, the following parameters are used, source domain = T (Training) and target domain = V (Validation), where source domain specifies what Visda2017 dataset domain to use for pretraining and target domain specifies what Visda2018 dataset domain to use for finetuning. We also train our model with a batch size of 32 and use GPU to speed up model training.

From training our "Ours2" and "Baseline 1" classification models, we obtain the results shown in Fig. 5.



	Baseline 1	Ours ($\lambda = 0, \gamma = 1$)
T -> V	83.27	83.05

Figure 5: Visda2017 Model Results

In the above result, the "Ours2" model has slightly higher accuracy than the "Baseline1" model. This validates our expectation that a model trained to ignore less relevant data will have higher accuracy compared to a model which does not do so.

To see if the accuracy values in our result are reasonable, we compare our results to the results from the original LBI paper for models trained on the Office31 and Officehome datasets. These are shown in Fig. 6 and 7.

	Baseline 1	Baseline 2	Baseline 3	Ours ($\lambda = 1, \gamma = 0$)	Ours ($\lambda = 0, \gamma = 1$)
A→W	99.38	96.88	97.5	97.5	**100**
A→D	95.83	96.88	97.92	95.83	**98.44**
D→W	98.12	96.88	98.12	98.12	**99.22**
D→A	**86.52**	84.38	83.79	85.55	85.94
W→D	98.96	96.88	98.96	97.92	**100**
W→A	**85.55**	85.16	85.16	85.35	85.35
Average	94.06	92.84	93.58	93.38	**94.83**

Figure 6: Original Office31 Model Results

	Baseline 1	Baseline 2	Baseline 3	Ours ($\lambda = 0, \gamma = 1$)	Ours ($\lambda = 1, \gamma = 0$)
Ar→Cl	65.28	66.09	**66.90**	68.15	66.67
Ar→Pr	85.70	84.98	84.62	85.94	**86.06**
Ar→Rw	71.76	70.60	73.15	72.00	**74.19**
Cl→Ar	57.08	58.75	61.46	60.49	**63.75**
Cl→Pr	85.22	84.86	85.22	85.70	**86.78**
Cl→Rw	70.60	71.30	73.50	70.43	**73.61**
Pr→Ar	59.38	58.54	59.79	61.38	**62.08**
Pr→Cl	**69.91**	66.55	67.36	69.11	68.87
Pr→Rw	71.18	70.49	72.69	72.60	**73.73**
Rw→Ar	59.79	60.83	**66.46**	64.06	65.63
Rw→Cl	**69.44**	65.05	69.21	70.31	68.75
Rw→Pr	86.78	84.62	**87.74**	87.38	86.66
Average	71.01	70.22	72.34	72.30	**73.07**

Figure 7: Original OfficeHome Model Results

We note that our model's accuracy is between that of the Office31 model and Officehome models. We also note that our "Ours2" model has a net accuracy improvement of 0.96% over our "Baseline 1" model. This improvement percentage is between the improvement percentage of the "Ours2" models compared to the "Baseline 1" models of the Office31 and Officehome datasets. Specifically, the Office31 and Officehome models in the original LBI paper have an improvement percentage of 0.8% and 2.9% respectively. From this comparison, we conclude that the accuracy values obtained for our model are reasonable and that LBI is able to improve classification accuracy for the Visda2017 dataset.

7 Discussion

For next steps, there are several avenues which we can explore. One such avenue is testing our current streetview recognition model with different domains to see if we can improve the accuracy of our model and to extend it to more classes such as traffic lights and road signs. This is with the aim that our model can classify a wider variety of street objects and so be able to be used in practical, real-life scenarios where there are a greater variety of obstacles which can be encountered.

Another interesting direction to explore is to test the Learning-By-Ignoring algorithm on other computer vision tasks besides classification such as object tracking, segmentation, and depth estimation to see if it can improve their performance. These are all tasks which are very important to current computer vision systems and so being able to improve them with our algorithm would be useful for many practical applications.

8 Conclusion

There are many applications in which classification is currently used both in the industry and research domains. Autonomous vehicles is one such application which has seen a lot of enhancement efforts made toward in recent years. One of the most important aspects of autonomous vehicles is that they be able to accurately sense objects in their surroundings whether through depth sensors like Lidar or optical means such as cameras. With camera information, autonomous vehicles need to be able to accurately classify and recognize surrounding objects in their surroundings. This is pivotal since incorrect classifications could endanger the occupants of a vehicle or bicyclists and pedestrians on the street. Improving the accuracy of classification methods is thus extremely important.

In our project, we show the Learning-By-Ignoring algorithm, which takes a "human-learning" approach to training classification models, improves recognition task accuracy. We do this by training two custom neural network models on the Visda2017 streetview dataset and giving evidence through our results that it has potential for use in training future autonomous vehicle recognition models. As autonomous vehicles become more and more common with the limited natural resources available in the world, we take steps towards making self-driving vehicles safer so that it can become a safer and more trusted form of transportation.

References

[1] Xingchen Zhao, Pentao Xie, “Interleaving-learning,” 2020. [Online]. Available: https://arxiv.org/pdf/2012.04863.pdf

[2] Xuefeng Du, Haochen Zhang, Pengtao Xie, “Learning by passing tests, with application to neural architecture search,” 2020. [Online]. Available: https://arxiv.org/abs/2011.15102

[3] Ramtin Hosseini, Pengtao Xie, “Learning by self-explanation, with application to neural architecture search,” 2020. [Online]. Available: https://arxiv.org/abs/2011.15102

[4] Xuefeng Du, Pengtao Xiee, “Small-group learning, with application to neural architecture search,” 2020. [Online]. Available: https://arxiv.org/abs/2012.12502

[5] Xingchen Zhao, Pentao Xie, “Learning by ignoring, with application to domain adaptation,” 2020. [Online]. Available: https://arxiv.org/abs/2012.14288

[6] Xingchao Peng, Ben Usman, Neela Kaushik, Judy Hoffman, Dequan Wang, Kate Saenko, “Visda: The visual domain adaptation challenges,” 2017. [Online]. Available: https://arxiv.org/pdf/1710.06924.pdf

[7] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” *arXiv preprint arXiv:1512.03385*, 2015.