COMP 551, FALL 2021, MCGILL UNIVERSITY

MiniProject 3: Multi-label Classification of Image Data

Arnaud Chol(261038600), Saad Benslimane(261031789), Elvin Zhang(261004488)

***Abstract*—We develop models to classify handwritten digits and english alphabet in combo MNIST dataset. By using ResNet34, data augmentation, and bagging, we achieve 96.07% test accuracy on Kaggle.**

I. INTRODUCTION

In this project, we train ResNet models to classify combo MNIST dataset. To reduce variance and avoid overfitting, we do data augmentation by denoising and transforming input images, and also use bagging for bootstrap aggregation. By these, we achieve 96.07% test accuracy. For unlabeled data, we utilize unlabeled data to train an Autoencoder model in order to denoise the input images, and to do label propagation to enlarge the training set.

II. DATASETS

Combo MNIST dataset has 30,000 labeled images and 30,000 unlabeled images for training. The dataset also comes with 15,000 test images. Each image is 56*56 and consists one English letter and one digit.

III. METHODOLOGY

A. Denoising

To images of the dataset were noisy. Multiple types of noise were contaminating the images : sparse high intensity noise in some images, and low intensity dense noise on some other images. To provide a clean dataset for the model to be trained on, we first computed a 3x3 average pooling combined with a threshold to either keep the value of the original image or set it to 0. This way, most of the noise was removed, and the letters and digits remained intact.

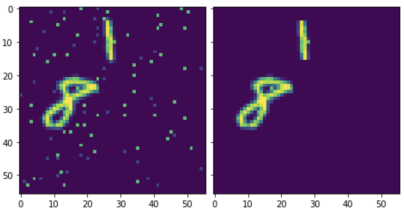

Figure 1: Original image and denoised image

B. ResNet

As a very first approach we tested using a really simple classifier made up of a few layers of convolutionnal filters. We rapidly found out that a much bigger model was needed because it produced a really low test precision of 40%.

We knew ResNet was probably our best bet. However it would requiere 112x112px images. To avoid unecessary computationnal computating, we tried to replicate the resnet architecture in a custom model that would take as an input 56x56px images. This model worked well and produced our first resonnable results of up to 87% precision while being computationnaly very efficient, training in a few minutes on a GeForce 2080Ti. However, much better scores were obtained when utilizing the full potential of the original ResNet model.

Residual Networks (ResNets) is a convolutional neural networks proposed by Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun in their 2015 computer vision research paper titled "Deep Residual Learning for Image Recognition".[1] ResNets builds on constructs known from pyramidal cells in the cerebral cortex, and do this by using "skip connections" to jump over some layers. Those models are implemented with double or triple layer skips that contain nonlinearities (ReLU) and batch normalization in between.

ResNets models have contributed immensely to the use of very deep neural networks; by limiting gradient loss in the deeper layers of it by adding a residual bond between each convolution layer.

The advantages of these networks are their ability to reduce gradient leakage in the lower layers of the network, and the possibility of their scaling at depth. The latter allows the architecture to be adjusted to the classification task at hand. Although costly in computing resources, they perform very well in classification on all levels of image detail [2][3]. Also we were able to train models in only 6 minutes, this enabled us to perform hyperparameter tuning, especially over the learning rate, the batch size and the weight-decay. We settled with a learning decreasing learning rate (0.0001 for the first 5 epochs, then 0.00001 for 5 epochs and 0.00005 for the last 5 epochs), batch size of 32 and weight decay of 0.003.

So the idea we used in this project was to label the unlabeled images using a ResNet34, and then combine the two datasets to train the same model ResNet34 before we classify the test dataset, Table 1 shows the results we got.

C. Data Augmentation

We quickly observed that our models were over fitting since the training accuracy often went up to 100% while the validation couldn't go past 82% (on our first models, ResNet34 did better). To reduce this overfitting, we decided to augment the dataset using rotation. Indeed, most of the digits were already rotated to some degree in the image, we figured that it would be a great way to increase the size of the dataset without risking croping the digits nor comprimising the meaning of each image provided that we used a low enough angle of rotation. From each image we were able to create 5, rotating from -45 to +45 degrees. We used scipy package to perform the rotations.

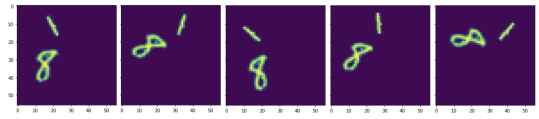

Figure 2: Data Augmentation

D. Bagging

Since we had plenty of prediction files because of the number of models we trained, we figured bagging would be a good attempt at merging all of those together. Our first naive approach was to simply make each model vote for the label of each test image. This method enabled us to get from 95% to 96% using our 5 best predictors. We later implemented a proper bagging by training the same model 40 times over 50% of the traning dataset. Bagging the models increased the accuracy from 92% to 95%, which was a good increase but not our top score. We then tested with 85% of the dataset for each model. Unfortunatly because the models must be so similar this way the final score didn't go over 95.5%.

If we had more time, we would have liked to implement a Bayesian bagging classifier that could make use of the fact that we know that each prediction file has a known approximate precision. This would have been a much clever method of combining those files together.

E. AutoEncoder

Autoencoders[4] use a neural network architecture to impose a bottleneck in the network which can compress knowledge representation of the original input. In this project, we first train them on unlabeled data. We add some white noise to unlabeled data prior to training, but compare the error to the original image when training. This forces the network from becoming overfitting to arbitrary noise in images. After training the encoder and decoder, we input the labeled data and generate denoised images.

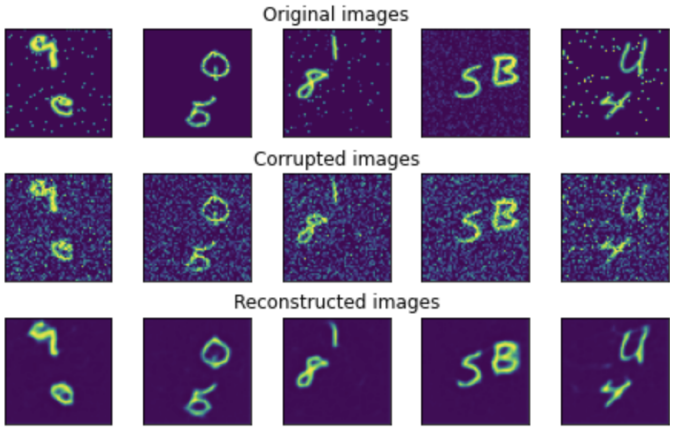

Figure 3: AutoEncoder denoising

We also tried to use autoencoder to directly predict the label of the images. This meant that we could feed it both labeled and unlabeled images. To do this, we added a loss to the previously mentioned reconstruction loss. This secondary loss concerned the bottleneck layer: the top 36 values were compared to the values of the label if the image was labeled using the binary cross entropy function. Also, if the label was provided with the image, the decoder was fed directly with the top 36 values replaced by the real label. This architecture however took a really long time to train (12h) and despite the reconstruction being quite good, the label prediction was only about 13% on training.

F. Label Propagation

Label Propagation Algorithm is a semi-supervised machine learning algorithms that assigns labels to unlabeled data observations in order to partition classification of data observations within dataset [5]. We thought this model would be useful to label the unlabeled images, so we did split the labeled images into train (80%) and validation (20%) dataset to test it's accuracy. Unfortunately the Label Propagation Algorithm did not classify the images as we wanted, after 1000 iterations its accuracy was 58% which is not enough to improve the accuracy of our ResNet model so we decided not to use it.

IV. Results

Figure 4 shows the typical training curve we obtained while training ResNet models, training accuracy gets to nearly 100% while the validation accuracy is left behind around 95%.

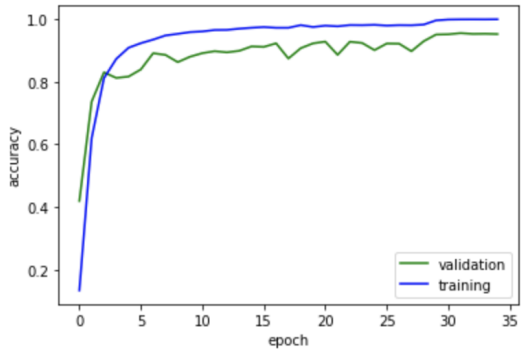

Figure 4: ResNet Accuracy Curve. The model overfits to the trainig data but gives reasonnable accuracy on validation set.

The following table compiling the main results we obtained during the assignment. CNN is the first convolutionnal simple model we tried, ResNet34 is the implementation of ResNet in pytorch. Resnet34x2 is the experiment we tried labeling the unlabeled data with a model before feeding it to a predictor to allow it to see more image diversity. Finally BAG means we used bagging to combine the results of multiple models.

Dataset	CNN	ResNet34	ResNet34x2	ResNet 34x2+BAG
Train data	93.5%	99,98%	99,96%	99,97%
Validation data	7.6%	95,6%	97,1%	97,1%
Test data	7.6%	95,55%	96,033%	96,066%

Table I: Accuracy results

From this table we can see that we reached our best accuracy when using the ResNet34 trained on the labeled dataset (with 95.55% accuracy on validation dataset) to predict unlabeled dataset, which improves the accuracy to 96.033%. The same model was retrained using the two datasets combined (labelead + unlabeled datasets) to predict the test dataset, using bagging we combined multiple models to achive 96.066% of accuracy.

V. Conclusions

The assignment allowed us to test multiple architectures of convolutional neural networks, from the simplest to the deeper ones. Testing with naive CNN first allowed us to really understand the benefits that residual network provide. We were also able to test multiple way of integrating unlabeled data, and despite the fact that none of them worked, we were able to create and train our own auto-encoders. We still have ideas of what could be done in an effort to improve the score. However, when looking at some of the data that is not correctly labeled, it is clear that even humans couldn't do much better, and asks the question of whether it would make sense to try to achieve better results or if it would just mean over fitting to a bit more data.

Statement of Contribution

1) Saad Benslimane: Implemented ResNet34, ResNet50 and ResNet152. Implemented the LabelPropagation algorithm to label the unlabeled images.

2) Elvin Zhang: Implemented ResNet18 and ResNet34. Trained an AutoEncoder Model on unlabeled data and used it to denoise labeled images.

3) Arnaud Chol: First worked on cleaning the data. Tried to implement a ResNet with 56x56 input size. Implemented the naive bagging script. Tried using autoencoder with label in bottleneck.

References

[1] Shaoqing Ren Jian Sun Kaiming He, Xiangyu Zhang. Deep residual learning for image recognition. 12:12, 2015.

[2] Linyao Shen Jia Huang Weiguang Sheng Qin Wang, Fengyi Shen. Lung nodule detection in ct images using a raw patch-based convolutional neural network. 2019.

[3] Negreeva A Tsukanova T Shifrin M Zakharova N Batalov A Pronin I Potapov A Danilov G, Kotik K. Classification of intracranial hemorrhage subtypes using deep learning on ct scans. 2020.

[4] Raja Giryes Dor Bank, Noam Koenigstein. Autoencoders. 12:22, 2020.

[5] Yannis Avrithis Ondˇrej Chum Ahmet Iscen, Giorgos Tolias. Label propagation for deep semi-supervised learning. 2019.