Semi-supervised learning for Diabetic Retinopathy

Hitesh Arora
hiteshar@andrew.cmu.edu

Alex Gaudio
agaudio@andrew.cmu.edu

Siddharth Satpathy
ssatpat1@andrew.cmu.edu

1 Introduction

Diabetic Retinopathy (DR) is a leading cause of blindness among working age adults. The disease generally has no symptoms until significant retinal damage has occurred. To detect presence of the disease, diabetic patients typically have regularly scheduled eye exams. The early stages of the disease are challenging for clinicians to identify in retinal fundus images [3]. Lesions such as micro-aneurysms and "spot and blot" hemorrhages can be just a few pixels in diameter, and are easily missed. The image analysis is also time consuming for clinicians. Integration of automated DR detection algorithms to assist physicians in DR diagnosis can improve sensitivity without a decrease in specificity [12].

Deep learning architectures, and in particular Convolutional Neural Networks, are well suited to general image analysis. Yet while most standard deep learning architectures are designed for large labeled datasets, medical image datasets are generally small, and labeling data is costly.

We propose a semi-supervised deep learning pipeline to predict the presence of DR. Our proposed architecture combines an auto-encoder with a classifier network. We introduce a loss function to enable concurrently training with unlabeled and labeled data. Our experiments show that a small amount of unlabeled data can improve performance of the classifier model by around 2%.

2 Background

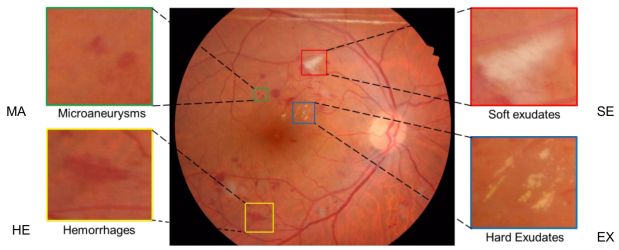


Figure 1: Fundus image presenting DR lesions: microaneurysms, hemorrhages, exudates (hard/soft).

Lesions indicative of DR include hard and soft exudates, microaneurysms, and hemorrhages. Hard Exudates (HE) are small accumulations of lipids and proteins. They appear in fundus images as bright, reflective dots in a circular pattern. Soft Exudates (SE) are ischemia of the Nerve Fiber Layer of the retina, and they appear as large white "cotton wool" spots. Microaneurysms (MA) are enlarged blood vessels; they appear as small, round red dots and are generally quite hard to see. MA are among the earliest recognizable lesions of diabetic retinopathy. Hemorrhages (HE), or bleeding, appear in various forms, and for DR they typically present small dots with less sharply defined edges. Figure 1 shows a selected retina image with microaneurysms, hemorrhages and exudates.

3 Related Work

Medical Image datasets are generally small and difficult to acquire and label. Methods that achieve good performance on DR detection are tailored to the task and the data available. Semi-supervised learning [1] and transfer learning [9] address the challenge of how to solve a learning task with an insufficient amount of data.

Semi-supervised learning (SSL) algorithms aim to improve the performance of supervised algorithms by also learning from unlabeled data. Studies have shown that semi-supervised methods trained on a dataset where some labels are ignored can approach [6] or even exceed ([14], [17]) performance of a supervised model trained on the fully labeled dataset.

Transfer learning (TL) aims to improve performance on a learning task by transferring knowledge gained on a related learning task. TL problems generally assume a dataset is split into two sets, where the sets are not necessarily independent and identically distributed. The assumption implies that learning algorithms can use datasets from varying data distributions to enhance performance on a particular dataset [9]. In particular, network based deep transfer learning is a widely used method for initializing model parameters by pre-training on a different dataset and learning task [16].

The work of [8] argues that much recent work in semi-supervised learning has limited applicability to real world data. Among the claims made, they find that SSL methods can degrade drastically when the unlabeled dataset contains out-of-distribution examples. Their results also suggest that transfer learning can in some cases outperform SSL techniques when a suitable dataset is available for pretraining. Third, they argue that realistic validation sets preclude reliable comparison of different hyperparameters. In the Discussion section, we consider these claims in the context of our work.

In [13], the authors develop a Stacked Sparse Autonencoder (SSAE) to perform binary classification on small image patches of retinal fundus images. Their approach requires patch-level annotations of retinal images, which are costly to acquire and less available than data with image-level labels. We extend their approach to DR detection by using image-level labels and by extending their network architecture to use deep convolutional layers within an autoencoder.

4 Methods

In this section, we describe the data set that we have use, the model architectures that we have developed, the loss functions that we use, data preprocessing and techniques that we used for evaluate our model.

4.1 Datasets

One of the largest publicly available datasets for DR is the Messidor dataset, which was put together by three hospitals in France with funding from the French Ministry of Research and Defense.

The Messidor dataset [2] is a high quality dataset of 1200 retinal fundus images captured between 2005 and 2006. Each image is given a DR grade from 0 (healthy) to 3 (Proliferative DR). While the images from each hospital were captured with the same brand of camera, the images from each hospital have different dimensions. Additionally, various errata exist in the data, including a few mislabeled images and several duplicate images. After addressing errata, we have 1187 images, 500 of which have DR grade 0 (healthy).

The Messidor Extension dataset [10] [2] is an additional set of 690 unlabeled images captured by the same Brest University Hospital used in Messidor data, from 2009-2010. We assume that the dataset shares a similar data distribution as the Messidor data. It is also among the largest available high quality fundus datasets. For these two reasons, it is a good choice as the unlabeled dataset.

4.2 Model Architecture

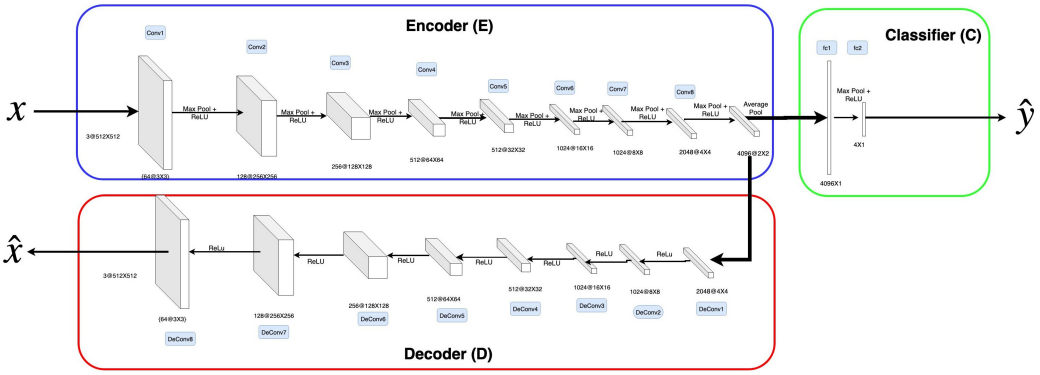


Figure 2: Semi-supervised deep network architecture. x, the input image, passes through Encoder and Decoder to generate reconstructed image, $\hat{x}$. x also passes through Encoder and Classifier to give a binary prediction $\hat{y}$. The above architecture describes our autoencoder classifier model (**AECL1**).

We approach the DR grading task as a semi-supervised binary classification problem by developing the architecture shown in Figure 2. The architecture utilizes unlabeled data to learn a robust feature representation which can be used to enhance performance of a classifier network.

The architecture has two main components: Classifier, C, and Autoencoder. The Autoencoder consists of an Encoder, E, and a Decoder, D. In particular, when given unlabeled data, x, we pass it through E and D to recover $\hat{x} \approx x$. When given labeled data, we pass x through E and C to predict a binary label $\hat{y}$, and we also pass x through E and D to recover $\hat{x} \approx x$. Training on both labeled and unlabeled data on this network allows us to update the encoder such that it learns features useful for both reconstruction and classification.

E is shared in common between the Autoencoder and Classifier. Since the primary goal of this architecture is to improve classifier performance with unlabeled data, we make a strong assumption that unlabeled data is always helpful to the classifier. Therefore, the majority of the classifier parameters are placed in E and a minimal number of parameters in C. The assumption implies that the data distribution of the unlabeled data is very similar to that of the labeled data.

The design of C enables integration with datasets of different kinds of labels, and C can have multiple outputs corresponding to different label types. In our designs, we ensure C has a minimum number of hyperparameters so that the majority of the classifier features are generated in E, and so they therefore also benefit by unlabeled weight updates. To transition from the encoder to the classifier, we first apply adaptive average pooling so the classifier network can work with images of different sizes assuming similar scale; adaptive average pooling guarantees the classifier input is always of predictable dimension for varying image dimension. C is fully connected, with a four unit layer followed by a single unit. The four units are added to enable future extension of the model to IDRiD dataset [11] for multi-class classification. The final single unit corresponds to a binary prediction.

We implemented two different architectures for semi-supervised learning model, which used the same C as explained above and different E and D as explained below. We refer to them as **AECL1** and **AERes18**.

AECL1 : As shown in Figure 2, **AECL1** is a symmetric autoencoder architecture where E has 8 convolutional layers and and D has 8 transposed convolutional layers. All convolution layers except the last deconvolution layer are followed by standard ReLU activations, max pooling and batch normalization layers.The advantage of having this symmetric architecture is that there is provision for weights of the layers in the encoder and decoder to be shared [4]. This can effectively reduce by half the number of parameters to optimize. **CL1** refers to the final classifier, i.e. E + C of **AECL1**.

AERes18: To leverage pre-trained weights, we designed a model based on ResNet18 ([5]), where E consists of the all the layers of ResNet18 except the last layer with pre-trained weights. D has

9 transposed convolutional layers without any skip connections with ReLU activation and batch normalization for each layer. **Res18** refers to the final classifier, i.e. E + C of **AERes18**.

Hereafter, we will use the symbol **AE** to refer to one of the two aforesaid model architectures, *viz.* AECL1 or AERes18.

4.3 Development of the Algorithm

We followed two approaches for training our model:

1. We use the fairly standard approach of [13] to pretrain the model on unlabeled data and then train it on labeled data. We first use only unlabeled data to train the auto-encoder (E and D) of the model to learn an encoding of the input based on reconstruction / autoencoder loss (L_{ae}). Then, we train the classifier part (E and C) of the model using only the labelled data based on cross-entropy / classification loss (L_{cl}). We call this training method "**AE then CL training**."

2. The second approach is training both the autoencoder and classifier parts of the model by feeding it minibatches that are either entirely labeled or entirely unlabeled. For an unlabeled minibatch, we train E and D using autoencoder loss, L_{ae}. For labeled data, we both use E and C to predict a binary label $\hat{y}$, and E and D to recover $\hat{x}$. In this case, we recover L_{ae} as well as a classifier losss, L_{cl}. To perform backprop, we combine the loss terms, $L_{combined} = f(L_{ae}, L_{cl}))$. To select between batches, we implemented a random sampling data loader that samples labeled and unlabeled minibatches without replacement. Minibatches are sampled based on the size of the respective labeled and unlabeled datasets. We discuss the motivation for this training method in 4.4. We call this training routine "**AE-CL combined training**." Figure 3 shows a plot of selection of labeled and unlabeled data in minibatches at different epochs.

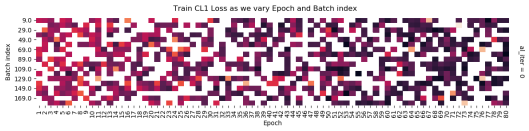


Figure 3: During training, we sample randomly between labeled or unlabeled minibatches based on dataset sizes. The white spots in this figure show regions where there was an unlabeled minibatch and the colored regions represent values of Cl1 losses for labeled data minibatches.

4.4 Loss Functions

We define loss terms for each corresponding output of our model.

The autoencoder loss, L_{ae}, we define as a simple MSE loss. Training with this loss helps the model generate smooth features [18] and learn the localized coloring of the fundus image. However, by design it does not preserve edge information very well, and a small change in alignment of two edges can result in a very large difference in magnitude.

$$L_{ae} = \frac{\sum_{i=1}^{n}(X_i - \hat{X}_i)^2}{n} \tag{1}$$

We discuss this loss term and possible alternatives that may be better suited to improve the classifier in the Discussion and future work section.

The classifier loss, L_{cl}, can be defined as a sum of classification loss terms corresponding to the outputs of C. More specifically, the number of terms in L_{cl} extends naturally to the number of labeled outputs available from the model. If training with just a single labeled dataset, such as Messidor with binary labels, standard binary cross entropy loss is appropriate and $L_{cl} = L_{\text{BCE}}$. If there were two labeled datasets, and the second had four binary classifications per image corresponding to lesion types indicative of DR (such as possible with the IDRiD dataset [11]), L_{cl} can naturally be expressed as a linear combination of loss terms. The coefficients in the linear combinatio n are discussed next.

Furthermore, in [7], an α term for cross entropy loss balances the weight of positive samples, providing a way to balance uneven class distribution. Since Messidor has an even balance of healthy

Table 1: This table contains the values of hyperparameters obtained when AERes18 is trained using *AE then CL training method* (columns 2, 3, 4, 5) and using *AE-CL combined training* (columns 6, 7, 8, 9). The first column represents percentile ranks for acccuracy values. Each row represents models for which accuracies are in the percentiles given in first column.

Percentile rank	**Learning rate** (L_{cl})	**Momentum** (L_{cl})	**Nesterov** (L_{cl})	**Weight decay** (L_{cl})		**Learning rate** (L_{combined})	**Momentum** (L_{combined})	**Nesterov** (L_{combined})	**Weight decay** (L_{combined})
0.75	5e-3	0.9	0.0 , 1.0	1e-5		5e-3	0.9	0.0	1.0
0.80	5e-3	0.9	0.0	1e-5		5e-3	0.9	0.0	1.0
0.85	5e-3	0.9	0.0	1e-5		5e-3	0.9	1.0	1.0
0.90	4e-3 , 5e-3	0.9	0.0	1e-5		5e-3	0.9	0.0	1.0
0.95	4e-3	0.5 , 0.9	0.0	1e-5		4e-3	0.9	1.0	1.0

and diseased images, we can assume that $\alpha = 1$. However, if we incorporate binary labels with class imbalance, the α can weigh positive classes differently than negative classes. This is important since in many medical contexts, most of the images captured from clinical screenings are healthy. If we were to input image patches, it is also likely the case that most patches of a diseased image are healthy.

$$L_{cl1} = \frac{1}{n}\sum_{i}^{n} \alpha y_{i,cl1} \log(\hat{y}_{i,cl1}) + (1 - y_{i,cl1}) \log(1 - \hat{y}_{i,cl1})) \tag{2}$$

Finally, we combine the unlabeled and labeled loss terms The inspiration for combined loss is to more effectively make use of latent features in the unlabeled data. Since the architecture has at least two loss functions, the loss terms will compete to influence the encoder weights. We hypothesize that the competition can improve the classification performance. However, if the magnitudes of the gradients $\frac{dL_{ae}}{dE}$ and $\frac{dL_{cl1}}{dE}$ are different, then each time parameters of the encoder are updated, the loss with larger gradients will dominate the weight updates. Therefore, we assume that the loss terms should be balanced, and we introduce a scalar λ for each classifier loss. For our project, we assume λ is a fixed scalar.

$$L_{\text{combined}} = L_{ae} + \lambda L_{cl} \tag{3}$$

4.5 Data Augmentation

All images were resized to 512×512 pixels and the following dataset augmentation was used during training: 15 degree random rotation, random scaling in the range [0.9, 1], and random horizontal flips.

4.6 Evaluating the Algorithm

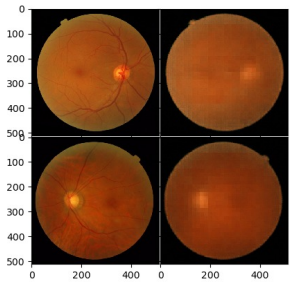


Figure 4: A selected image where we show reconstruction of retinal images. In the left column, we show the original images and in the columns on the right, we show reconstructed images.

All experiments use the Messidor dataset as the fully labeled dataset. The Messidor Extension is our unlabeled dataset. In this work, we assume the binary labels of healthy (DR grade = 0) or diseased (DR grade > 0). Reducing the task to a binary classification balances the dataset and simplifies the

problem and evaluation. In particular, because the classes are balanced, we found accuracy and area under the ROC curve to be equivalent in all experiments.

For both datasets, we apply a 60/20/20 random split stratified across hospitals. The train and validation split is re-computed each time a model is trained, but the test set is always the same set of 247 images. Where test accuracy is reported, we train the model on the combined training and validation sets.

5 Results

5.1 Grid Search

Hyperparameters for Res18 were validated via grid search over a four dimensional parameter space: learning rate, classical and Nestorov momentums and weight decays. We explore four different values of learning rates ($[0.002, 0.003, 0.004, 0.005]$), three separate values of momentums ($[0.1, 0.5, 0.9]$) and four values of weight decays $[10^{-7}, 10^{-5}, 0.01, 0.1]$. Table 1 gives modal values of learning rates, classical momentums, Nesterov momentums and weight decays for 'AE then CL' (L_{cl}) and 'AE-CL combined' (L_{combined}) training for samples selected from different percentile ranks of accuracy values.

5.2 Baseline Implementation Details

We trained 4 baseline models.

InceptionV3 was trained with a batch size of 2 using Adam optimizer with a learning rate of 2×10^{-4}, weight decay of 0.01, default betas (0.9, 0.999) and an epsilon of 0.1. We found that 0.1 was necessary to improve stability of the model. The small batch size was necessary to fit the model in our GPU. The parameters for this model were chosen from [15], and we add an epsilon to stabilize the model.

Resnet18 was trained with a batch size of 48 using SGD optimizer with a learning rate of 0.001, weight decay of 0.01, and Nesterov momentum of 0.9. The parameters for this model were chosen from [14].

CL1 was trained with a batch size of 6 using Adam optimizer with a learning rate of 5×10^{-6}, weight decay of zero, betas of (0.95, 0.999) and epsilon of 1×10^{-6}.

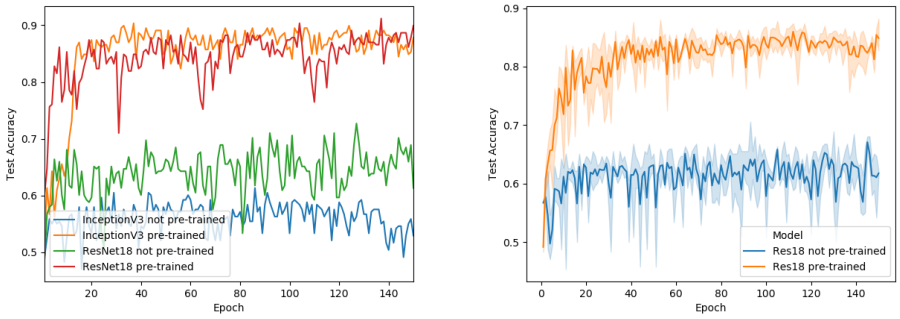
Res18 was trained with a batch size of 48 using SGD optimizer with a learning rate of 0.004, weight decay of 1×10^{-5}, and (non-Nesterov) momentum of 0.9. These parameters were chosen via grid search on the validation set. The details of the grid search are described in section 5.1.

5.3 Pre-training improves performance and stability

We first evaluate InceptionV3 and ResNet18 models on the Messidor data, and we observe how the performance of these baselines changes with and without pre-initialized weights on ImageNet. Figure 5a shows the test set accuracy over epochs. Test set accuracy for InceptionV3 and ResNet18 improves with pretraining by approximately 30% and 20%, respectively. The results clearly show that pretraining on ImageNet improves binary classification performance on Messidor. This is interesting because ImageNet and Messidor have different data distributions; ImageNet has natural images grouped into 1000 classes, while Messidor has retinal fundus images with binary labels. Pre-training on a dataset with more similar distribution may improve the performance even further.

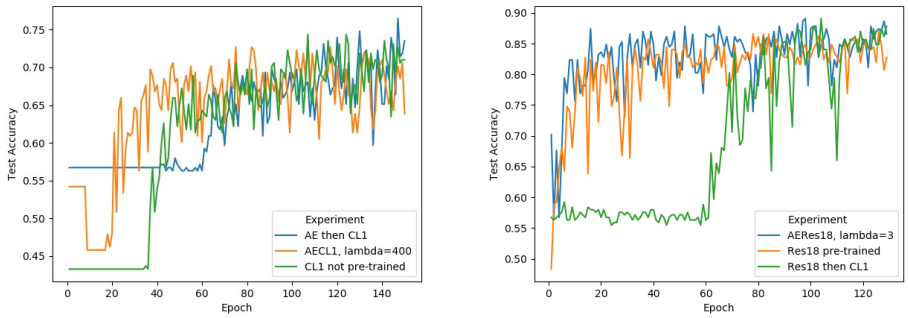
Moreover, the variance in test accuracy over epochs decreases after pretraining for both ResNet18 and Inception. Therefore, the results show that pretraining leads to more accurate models.

We also note that InceptionV3 performs worse than ResNet18 when it is not pre-trained. InceptionV3 is a more complex model, with 27,161,264 parameters compared to Resnet18's 11,689,512 parameters. Since Messidor has a total of 747 million pixels in the 950 training images, the ratio of pixels to parameters is 27.5 and 63.9, respectively. As a comparison, a dataset like ImageNet has on the order of 1,281,167 images, and the ratio of pixel to parameter ratios for the two models is much larger in this case. The ratios on Messidor underscore the importance of regularization, since overfitting becomes an issue as the number of parameters approaches to the amount of data.



(a) **InceptionV3 and ResNet18 baselines.** (b) **Our Res18 baselines.**

Figure 5: Our Res18 model shows similar performance to the standard ResNet18 and InceptionV3 models.



(a) **CL1** (b) **Res18**

Figure 6: Performance of CL1 and Res18 models for 3 different training methods.

We also evaluate performance of our Res18 classifier. Figure 5b shows our Res18 baseline with and without pretraining. Res18 is essentially ResNet18 with top layers defined by C, as described in the Methods section. Figure 5 shows that the two models attain similar performance.

5.4 Unlabeled data improves classification performance

We conducted several experiments using this architecture to evaluate how the incorporation of unlabeled data improves performance.

Looking at AE then CL1 curve in Figure 6a shows that just pre-training on unlabeled data using autoencoder helps shown by the higher starting accuracy. The flatter accuracy line until 60 epochs shows that some of the AE learned features need to be relearned for classification task. AE then CL1 has higher max accuracy than CL1, showing an accuracy improvement of **1.68%** over the CL1 baseline.

Looking at AECL1 curve in Figure 6a shows the combined loss model converges faster than AE then CL1 and CL1 models. Its max accuracy isn't higher than the CL1 baseline, suggesting the fixed λ method needs room for improvement which we address in future work.

Table 2: AERes18 and AECL1 Accuracy results

Model	**Baseline max accuracy**	**AE then CL max accuracy**	**AE-CL combined max accuracy**
AECL1 ($\lambda = 400$)	74.79	76.47	73.53
AERes18 ($\lambda = 3$)	86.97	89.08	89.08

Looking at the AERes18 curve in 6b shows the combined model has higher max accuracy than Res18, showing an accuracy improvement of **2.1%** over the Res18 baseline.

6 Discussion and future outlook

Using semi-supervised learning on AERes18 with pre-trained weights, we obtain 2.1 % higher max classification accuracy than ResNet18 classifier with pretrained weights. In our work, we have been able to show that it is possible to achieve improvements over existing state-of-the-art baseline methods like ResNet18 using semisupervised classification methods.

At the same time, we are interested in testing different loss functions in our model in a future version of our work. More specifically, we intend to try KL-divergence and mutual information loss. These loss functions consider distributions around pixels of images. Signals of DR like microaneurysms, hemorrhages and hard and soft exudates exist locally in retina scans. Hence, KL divergence and mutual information loss are likely to give us more accurate reconstructions which preserve this local pixel level information.

[8] found that SSL algorithms are sensitive to varying amounts of data. It would be worth using unlabeled data from other datasets such EyePACS which we recently discovered.

The model could be extended to solve a multi-class classification and predict to predict DR severity problem, which is why we designed fully-connected layer with 4 units.

An idea is to incorporate active learning sampling method to choose unlabeled data to and explore whether the method can enable our architecture to incorporate datasets of different data distributions.

As we explained earlier that during simultaneous training, if the magnitudes of the gradients $\frac{dL_{ae}}{dE}$ and $\frac{dL_{cl1}}{dE}$ are significantly different, then the loss with larger gradients will dominate the encoder weight updates. Therefore, we introduce a scalar λ, such that $L_{\text{combined}} = L_{ae} + \lambda L_{cl}$ to balance gradients and have experimented with a fixed λ. A potential extension is to have an adaptive λ where

$$\lambda_{t+1} = ||\frac{(\frac{dL_{ae}}{dE})_t}{(\frac{dL_{cl}}{dE})_t \lambda_t}|| \tag{4}$$

References

[1] O. Chapelle, B. Schölkopf, and A. Zien, editors. *Semi-Supervised Learning*. MIT Press, Cambridge, MA, 2006.

[2] E. Decencière, X. Zhang, G. Cazuguel, B. Lay, B. Cochener, C. Trone, P. Gain, R. Ordonez, P. Massin, A. Erginay, B. Charton, and J.-C. Klein. Feedback on a publicly distributed database: the messidor database. *Image Analysis & Stereology*, 33(3):231–234, Aug. 2014.

[3] T. F. Farley, N. Mandava, F. R. Prall, and C. Carsky. Accuracy of primary care clinicians in screening for diabetic retinopathy using single-image retinal photography. *The Annals of Family Medicine*, 6(5):428–434, 2008.

[4] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[5] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[6] S. Laine and T. Aila. Temporal ensembling for semi-supervised learning. *CoRR*, abs/1610.02242, 2016.

[7] T. Lin, P. Goyal, R. B. Girshick, K. He, and P. Dollár. Focal loss for dense object detection. *CoRR*, abs/1708.02002, 2017.

[8] A. Oliver, A. Odena, C. A. Raffel, E. D. Cubuk, and I. Goodfellow. Realistic evaluation of deep semi-supervised learning algorithms. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 3235–3246. Curran Associates, Inc., 2018.

[9] S. J. Pan and Q. Yang. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359, Oct 2010.

[10] G. Quellec, M. Lamard, P. M. Josselin, G. Cazuguel, B. Cochener, and C. Roux. Optimal wavelet transform for the detection of microaneurysms in retina photographs. *IEEE Trans Med Imaging*, 27(9):1230–1241, Sep 2008. PMC2567825[pmcid].

[11] P. P. S. P. R. K. M. K. G. D. V. Sahasrabuddhe and F. Meriaudeau. Indian diabetic retinopathy image dataset (idrid), 2018.

[12] R. Sayres, A. Taly, E. Rahimy, K. Blumer, D. Coz, N. Hammel, J. Krause, A. Narayanaswamy, Z. Rastegar, D. Wu, S. Xu, S. Barb, A. Joseph, M. Shumski, J. Smith, A. B. Sood, G. S. Corrado, L. Peng, and D. R. Webster. Using a deep learning algorithm and integrated gradients explanation to assist grading for diabetic retinopathy. *Ophthalmology*, 126(4):552–564, Apr 2019.

[13] J. Shan and L. Li. A deep learning method for microaneurysm detection in fundus images. In *2016 IEEE First International Conference on Connected Health: Applications, Systems and Engineering Technologies (CHASE)*, pages 357–358, June 2016.

[14] A. Smailagic, P. Costa, A. Gaudio, K. Khandelwal, M. Mirshekari, J. Fagert, D. Walawalkar, S. Xu, A. Galdran, P. Zhang, A. Campilho, and H. Y. Noh. Online active learning for medical image analysis. unpublished, N.D.

[15] A. Smailagic, H. Y. Noh, P. Costa, D. Walawalkar, K. Khandelwal, M. Mirshekari, J. Fagert, A. Galdran, and S. Xu. Medal: Deep active learning sampling method for medical image analysis. *CoRR*, abs/1809.09287, 2018.

[16] C. Tan, F. Sun, T. Kong, W. Zhang, C. Yang, and C. Liu. A survey on deep transfer learning. *CoRR*, abs/1808.01974, 2018.

[17] V. Verma, A. Lamb, J. Kannala, Y. Bengio, and D. Lopez-Paz. Interpolation consistency training for semi-supervised learning. *CoRR*, abs/1903.03825, 2019.

[18] H. Zhao, O. Gallo, I. Frosio, and J. Kautz. Loss functions for neural networks for image processing. *CoRR*, abs/1511.08861, 2015.