Guiding the management of cervical cancer with convolutional neural networks

Stephen Pfohl
Stanford University
spfohl@stanford.edu

Oskar Triebe
Stanford University
oskar2@stanford.edu

Ben Marafino
Stanford University
marafino@stanford.edu

Abstract

In order to appropriately treat cervical cancer, making an accurate determination of a patient's cervical type is critical. However, doing so can be difficult, even for trained healthcare providers, and no algorithms currently exist to aid them. Here, we describe the development and implementation of a convolutional neural network-based algorithm capable of distinguishing between three cervical types. Our approach extends standard transfer learning pipelines for fine-tuning a deep convolutional neural network, specifically a residual network (ResNet), with customized data augmentation and model ensembling techniques. Our algorithm achieves an accuracy of 81% *on our internal test set and a multi-class cross-entropy loss of 0.557 on the Kaggle test set, resulting in a leaderboard position of 25th out of over 800 teams, as of June 12, 2017, with three days left to finalize model submissions.*

1. Introduction

Cervical cancer is the fourth most common cancer worldwide, with roughly 600,000 new cases and 300,000 deaths annually [1]. While most cases of cervical cancer can be prevented with timely screening, and even cured with appropriate treatment, selecting the most effective treatment depends on the anatomical type of a patient's cervix – in particular, the type of transformation zone (TZ) of their cervix. Under the current classification, there exist three types of TZs (1, 2, and 3), which can be distinguished from each other visually, via coloposcopy [2].

Accurate determination of a patient's TZ type is critical in order to guide appropriate treatment. Indeed, a physician who incorrectly classifies a patient's cervix and then performs an inappropriate surgery may fail to completely remove the malignancy, or may even increase their patient's future cancer risk by forming scar tissue that obscures future cancerous lesions [2]. However, determining the type of a patient's TZ from a cervigram image can be difficult, even for trained healthcare providers using colposcopy, and no computer vision-based algorithms or classifiers currently exist for this problem. Such an algorithm could be used for clinical decision support, thus greatly facilitating providers' workflows, especially in rural settings and in developing countries where resources are scarce.

Using data released as a part of a Kaggle competition [3], we aim to create a convolutional neural network-based algorithm to classify cervical TZ type from cervigram images. Our approach, given the limited availability of these data, is based on transfer learning [4–7], wherein we fine-tune to our data a type of deep neural network, specifically a *residual network* (ResNet) [8] pre-trained on the ImageNet classification task [9].

2. Related work

Deep learning, and convolutional neural networks [10] (CNNs) in particular, are as of late increasingly finding wide application in the field of medical image analysis. Specific applications of these algorithms include classification [11–13], object detection and anatomical localization [14–16], segmentation [17–19], registration (i.e., spatial alignment) [20], retrieval [21], and image generation and enhancement [22].

In medical image classification, the inputs are often one or more photographs or scans (CT, MRI, PET, etc.) belonging to a patient and the output is some kind of disease or anatomical status associated with that patient. For example, retinographs can be used to stratify a patient with diabetic retinopathy into one of seven or eight grades corresponding to the severity of their disease [11].

CNN-based approaches to medical image classification problems have taken advantage of two transfer learning strategies which predominate in the literature: 1) fine-tuning a CNN pre-trained on a large, generic image dataset, such as ImageNet; or 2) using a similarly pre-trained CNN, but as a *fixed feature extractor* (FFE), by freezing all but its last FC layer before tuning it to the data corresponding to the task of interest. These approaches are motivated by the dearth of the large volumes of task-specific medical image data required to robustly train a CNN from scratch. While many image classification tasks in the medical domain are more narrowly focused in scope compared to the ImageNet task, the generic features learnt by networks trained on ImageNet appear to generalize well to medical and other image classification tasks.

There appears to be no consensus in the literature as to whether fine-tuning a CNN outperforms CNN-as-FFE in this domain. Indeed, the few currently extant papers that do report the results of both approaches [23, 24] have reached conflicting conclusions, with [23] finding that approach (1), fine-tuning, outperforms approach (2), CNN-as-FFE, in classification accuracy, and vice versa in [24]. Notably, recent landmark papers in the medical image classification field [11, 13], have relied on fine-tuning a pre-trained CNN based on the Inception-v3 architecture [25], achieving near-expert performance on diabetic retinopathy and melanoma classification tasks. Fine-tuned, pre-trained Inception-v3 CNNs have also been used to classify and detect metastases on gigapixel breast cancer pathology images [12].

However, CNNs have yet to be widely applied to image classification tasks specifically motivated by cervical cancer. Standard feedforward networks have been used to detect abnormal cells in Pap smears, with the aim of assisting cervical cancer screening [26]. More recently, cervigrams have been classified and segmented using classical machine learning methods, such as SVMs relying on PHOG features [27] and KNNs [28]. In two cases, CNNs have been applied to cervigram analysis: [29] and [30] used a pre-trained AlexNet [31] to classify cervigrams in order to aid the diagnosis of cervical dysplasia. However, no prior work that investigates the problem of cervical TZ classification appears to currently exist. We note that, in contrast with much past work in the cervical image analysis domain, however, the end goal of this task is not necessarily to aid diagnosis, but to help guide *treatment* by enabling healthcare providers to make more accurate determinations of a patient's cervix type in order to select the appropriate treatment of their cervical cancer.

3. Dataset

3.1. Overview

The data are drawn from the Intel/MobileODT colposcopy image database, which contains 8,215 labeled images, in addition to an unlabeled leaderboard test set of 512 images. These 8,215 images, which we used to form the train/validation/test data splits, can be divided between a *base* set, consisting of 1,481 images, and an *additional* set of 6,734 images. There are three classes corresponding to the three types of cervical TZs, and their distribution is given in Table 1. Nearly all images in the dataset are present in one of three resolutions: 2448×3264, 4160×3120, or 4128×3096. All images are in `.jpg` format and in color using a RGB color profile.

The cervigram images in the dataset depict the cervix as viewed through a colposcope, and so tend to be magnified and fully illuminated. Exemplars representing each of the three types of TZs are shown in Figure 1. The region of interest corresponding to the TZ is in the center of the image, circumscribing the opening (external cervical os) that connects to the uterus. In many images, the speculum used to dilate the vagina (so as to aid visualization) is also visible, and in some cases, it partially occludes parts of the cervix. There is also considerable variation in the extent of vignetting, which can be seen in Figure 1.

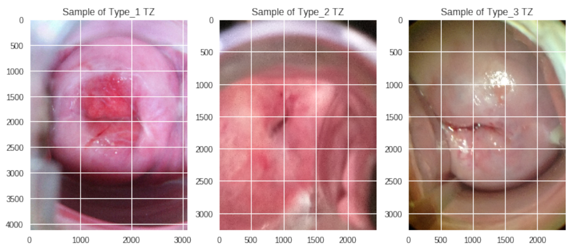


Figure 1: Exemplars of the three types of cervical TZs represented in the dataset.

	Type 1	Type 2	Type 3
Base train set	250	781	450
Additional train set	1,191	3,567	1,976
Total	8,215 images		
Internal train set	30% base + 100% additional		
Internal validation set	50% base		
Internal test set	20% base		
External test set	512 images, unlabeled		

Table 1: Distribution of the three classes in the dataset. Note that the percentages in this table refer to our alternate data splitting strategy; refer to §3.3 for more details.

3.2. Data noise and heterogeneity

In order to characterize the extent of heterogeneity present within our dataset, we initially performed an exploratory data analysis of the images, focusing principally on clustering the images using t-SNE [32] to characterize the extent of heterogeneity present in the data. An example output from a t-SNE run is presented in Figure 2; due to computational considerations, and to facilitate visualization, we worked with relatively small samples of data. We were able to identify several sources of heterogeneity in the data via our EDA. In particular, these sources include 1) image duplication; 2) non-cervix images; and 3) variation in appearance due to staining of the cervix, e.g. with iodine or acetic acid.

We found that duplicate images tended to occur via one of two mechanisms: either multiple images were taken of a single patient's cervix as a part of the same examination, albeit in slightly different poses; or the cervix was re-imaged using a green filter attached to the colposcope. Examples of these types of duplication can be seen in Figure 2, particularly in the northwest quadrant of the figure. It is worth noting that the presence of duplicate images was mostly limited to the additional training data set rather than the base train set.

We initially believed that the green images were the result of data corruption or some other abnormal process, but we found that using the green filter allows the doctor to more easily visualize malignancies and other lesions. As such, while we had considered excluding these images from the data, we now recognize these as plausible input to the algorithm.

Our EDA also identified non-cervix images, including images of what appear to be some kind of plastic drape, a Motorola logo, and even of a face – which can be seen towards the right side of Figure 2. These non-cervix images – five in total – were manually removed from our dataset. Finally, we also identified images where the cervix appeared to be stained with either acetic acid or an iodine solution – techniques used to aid the visualization of lesions.

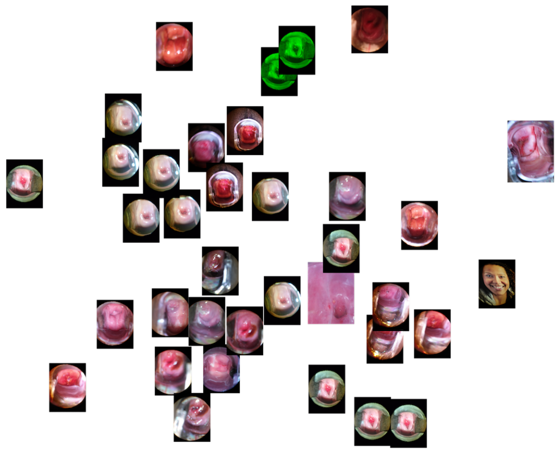


Figure 2: Results of t-SNE embedding of a sample of the data, performed as a part of our exploratory data analysis, indicating significant heterogeneity.

3.3. Dataset splitting

Originally, all 8,215 images were partitioned to form the initial split of the data into train, validation, and test sets in a 70:20:10 ratio. However, we found that duplicate or near-duplicate images from the same patient tended to predominate in the "additional train set" (Table 1), which was provided separately from the "base train set" by Kaggle. With a naive partitioning of the data, we found that these duplicate images were present in either both the train and validation sets, or the train and test sets, representing an example of data leakage that afforded overly optimistic estimates of the true Kaggle test set loss. Over the course of our experiments, we investigated an alternative split which partitioned the provided data such that the internal validation and test sets contained only images from the "base train set" (see Table 1). This change of data split was motivated principally by a desire to obtain more reliable estimates of the Kaggle test loss.

In this paper, we refer to this alternative split as the **alternate data split** (ADS), and consists of a split that partitions the "base train set" of 1,481 images into train, validation, and test sets in a 30:50:20 ratio, and all 6,734 images in the "additional train set" were allocated to the train set. Unless otherwise specified, all results presented correspond to those obtained with the alternate data split, as we found that such results were more representative of performance on the Kaggle test set.

4. Methods

4.1. Overview

Our approach to the problem of cervix TZ classification relies principally on transfer learning, using 18- and 34-layer deep residual networks (ResNets) [8, 33]. ResNets resemble deep convolutional neural networks in overall architecture, but with one key difference being the presence of *skip connections* between blocks of convolutional layers (Figure 3). These skip connections enable the blocks in the network to learn the residual mapping $\mathcal{F}(\boldsymbol{x}) = \mathcal{H}(\boldsymbol{x}) - \boldsymbol{x}$ for some input $\boldsymbol{x}$, rather than the full mapping $\mathcal{H}(\boldsymbol{x})$. The input $\boldsymbol{x}$ is simply carried forward, unmodified, through the skip connection and added to the output of the block, $\mathcal{F}(\boldsymbol{x})$. While both maps are asymptotically approximately equivalent [8], the residual mapping may prove easier for the network to learn than the identity. While this subtle insight has proven fruitful, recent evidence has shown that ResNets appear to behave like ensembles of exponentially many relatively shallow networks [34]. This behavior could explain the performance characteristics of ResNets and their relative ease of training – the latter which has already in part been explained by their avoiding the vanishing gradient problem encountered in deep networks, thanks to these skip connections. The architectures of the two ResNets we tested in our experiments are shown in Table 2.

All methods were implemented using PyTorch [35, 36] and trained on 1 to 4 $\times$ NVIDIA Tesla K80 GPUs on a Google Cloud virtual machine.

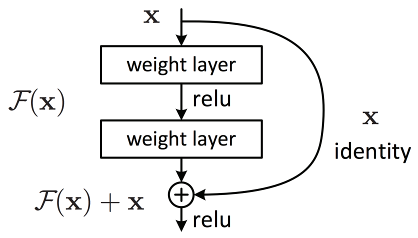


Figure 3: Schematic of the basic building block of a ResNet. Note the "skip connection" that preserves and adds the identity map to the output $\mathcal{F}(\boldsymbol{x})$ of the block to obtain $\mathcal{F}(\boldsymbol{x})+\boldsymbol{x}$. Figure taken from [8].

			ResNet-	
Block	Output size	Layer(s)	18	34
`conv1`	112×112	[64 7×7 filters] $\times$	1	1
`conv2`	56×56	3×3 maxpool $\times$	1	1
		$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times$	2	3
`conv3`	28×28	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times$	2	4
`conv4`	14×14	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times$	2	6
`conv5`	7×7	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times$	2	3
avgpool, FC ($1000 \rightarrow 3$), softmax				

Table 2: Architectures of the two ResNets tested. The first layer in each `conv*` block (including `conv1` itself) performs downsampling with a stride $S = 2$. Batch normalization [37] is also applied to the output of each convolutional layer, and ReLU nonlinearities [38, 39] are also then applied to the batchnormed output of each convolutional layer in each block. The skip connection incoming into each block is added to the output of the second convolutional layer, right before the second ReLU nonlinearity (see Figure 3). Modified from [8].

4.2. Image preprocessing

4.2.1 Basic preprocessing

All images were scaled off-line such that the smaller dimension had length 256 pixels. This significantly sped up training time, due to the lower computational cost incurred. Our basic image preprocessing pipeline at train time proceeds as follows:

1. First, training images were cropped at a random location to extract a 224×224 region.
2. We then applied a random horizontal flip with probability 0.5.
3. Finally, the images were normalized channel-wise, using the ImageNet statistics, to have mean±standard deviation 0.485 ± 0.229 for the red channel; green, 0.456 ± 0.224; and blue, 0.406 ± 0.225.

At test time, the images were pre-scaled to have the length of their smaller dimension be 224 pixels, and then center cropped. For step (3), we also experimented with dataset-specific normalization statistics, which were computed based on the images in train set, but found no significant variation in validation or test performance.

4.2.2 Augmented preprocessing

As a form of regularization, we experimented with a more comprehensive set of transforms, described below in the data augmentation techniques. Applying these transforms to the images in each training minibatch at train time was computationally expensive, increasing the time per epoch by a factor of two or more, even when applied to pre-scaled images as in the basic preprocessing strategy.

To allow for faster training, the images were instead transformed in advance and stored as a separate, augmented dataset. Each image was loaded, transformed and saved 10 times for images in the main training set, and 3 times for images from the additional dataset. All transforms were applied randomly and independent of each other to each image. As a code basis, we used the `torchsample` library for PyTorch [40], in addition to the standard PyTorch `transforms`, and then modified and extended it in order to accommodate the types of augmentations desired.

Data augmentation strategy

- **Pad** the long edge by 20% on each side and the short edge to match the long edge, resulting in a square image;
- **Flip**, either horizontally or vertically, or both, with both types of flips having independent probability 0.5;
- **Rotation**, at random, by an angle $-180 \leq \theta \leq 180$ degrees;
- **Translation**, at random, by up to 5% in either dimension;
- **Zoom**, into the image, randomly cropping the center 0.45 to 0.80 region;
- **Scale**, to a square 224×224 image.

Finally, a center cropped 224×224 version of the non-augmented data was added to the augmented dataset in order to avoid overfitting to the transformed data. The train time preprocessing pipeline for augmented images consisted of:

1. First, the 224×224 images were padded by 16 pixels on all sides.
2. Next, training images were cropped at a random location to extract a 224×224 region.
3. We then applied a random horizontal and vertical flip with probability 0.5, each.
4. Finally, the images were normalized channel-wise, as in the basic preprocessing step (3).

4.2.3 Data ensembling

As deep convolutional networks may learn to distinguish a certain object type only under certain image conditions, unrelated to the actual physical properties of the object, we experiment with a technique that we call "data ensembling" to minimize the impact of such effects. Under this technique, we the apply the data augmentation strategy, outlined in §4.2.2, to ten replicates of the test images and then pool the resulting augmented images with center-cropped versions of original test images. Thereafter, we classify each of the 11 resulting images and average the class label probabilities over the images having the same origin image. An additional goal of this technique is the smoothing of overconfident model predictions.

4.3. Experimental conditions

Our main experimental conditions are summarized as follows:

- **ResNet-18, random initalization** (ResNet-18-RI)
- **ResNet-18, fine-tuned** (ResNet-18-FT)
- **ResNet-34, random initalization** (ResNet-34-RI)
- **ResNet-34, fine-tuned** (ResNet-34-FT)

We define **fine-tuning** to be a transfer learning procedure wherein the weights of a pre-trained network are loaded, the output projection layer replaced with one of the correct size for the new task, and then the entire network trained as usual. For random initialization, we simply train the entire network from scratch without loading any weights of a pre-trained model. Using the pre-trained network as a fixed feature extractor was briefly explored, but we abandoned this approach as these models appeared to lack sufficient representational capacity for cerical TZ classification.

For the four architectures and choices of initalizations above, we initially performed a coarse random hyperparameter search [41] over the learning rate and the L_2 regularization constant λ. We then expanded our experiments to include step-based learning schedules, as well as to investigate finer random grids of learning rate and λ. We also investigated other architectures, including SqueezeNet [42] and Inception-ResNet [43]. In total, we ran over 200 experiments.

For all networks, we used the Adam optimizer [44] with $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 1 \times 10^{-8}$. We used a minibatch size of 64 in all experiments. We also tested an optimizer that used SGD with Nesterov momentum [45, 46], but found that it consistently yielded slower convergence compared to Adam.

4.4. Model ensembling

Combining classifiers in an ensemble to create predictions is a long-used technique in machine learning [47, 48] to produce more stable and less overly confident estimates of class probabilities. Recently, with the increasing availability of cheap compute, ensembles of CNNs appear to be coming into vogue, especially in the medical image analysis field [49, 50]. Label smoothing (as in §7 of [51]) or a maximum-entropy based confidence penalty [52] can also be used to regularize model outputs, but we chose not to investigate them further at this time.

With this in mind, we also tested the following collection of model ensembles, with the aim of improving the generalization of our models and ultimately the performance on the Kaggle test set. We also computed performance metrics (detailed in the next subsection) for each ensemble on the test set.

First, we define the **Mean-of-top-K-models ensembles**. Under this strategy, the top K models that achieve the lowest validation loss are used to independently make predictions on the test set. For each image in the test set, the mean of the predicted probabilities for each class across the K models are used as the resulting predicted probabilities for the ensemble. The composition of the ensembles investigated further are as follows:.

- $K = 5$: $3 \times$ ResNet-18-FT. $2 \times$ ResNet-34-FT.
- $K = 7$: $5 \times$ ResNet-18-FT, $2 \times$ ResNet-34-FT.
- $K = 10$: $8 \times$ ResNet-18-FT, $2 \times$ ResNet-34-FT.

We additionally explore the idea of a **Diverse Ensemble**, which we define as a mean ensemble constructed as above with the highest performing single model of a diverse set of model classes (in this case, the models listed in the upper section of Table 3).

4.5. Evaluation metrics

In addition to monitoring model performance over the course of each experimental run, we also saved the best-performing model checkpoints from each run and used them to generate predictions and performance results on the internal test set, as well as on the external Kaggle test set. Following hyperparameter tuning, we do not retrain the model with the combined training and validation set and instead directly utilize the checkpoint obtained during training.

The classification results of our models are evaluated principally using the multi-class log loss (or cross-entropy [CE] loss):

$$\text{logloss} = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{M} y_{ij}\log(p_{ij}),$$

where N denotes the number of images in the test set, M the number of classes, y_{ij} the ground truth label that image i belongs to class j, and p_{ij} the predicted probability of image i belonging to class j. As a part of the model development process, we tracked this CE loss and used it to compare models and tune hyperparameters. We also computed class-level F_1 scores for prediction on the internal validation set. The F_1 score is defined as

$$F_1 = 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}},$$

and can be interpreted as a weighted mean of precision and recall that weighs both equally. Finally, we also computed and tracked the overall accuracy across all three classes of each of our models.

5. Results

5.1. Hyperparameter tuning experiments

We carried out the random hyperparameter search procedure, as previously described in §4.4, to train over 200 models of various architectures, primarily ResNet-18 models and ResNet-34s. The distributions of the minimum validation loss achieved over the course of each experiment for each condition (model × initialization) are shown in Figure 4. Overall, these results indicate that the fine-tuning (FT) approach outperforms training from a random initialization (RI), and that training with the use of data augmentation appears to reduce generalization performance for both the ResNet-18 and the ResNet-34 models. Qualitatively, the centroids of the distributions of validation loss for the ResNet-18-FT and the ResNet-34-FT procedures are roughly equal, but the ResNet-18 appears to achieve validation losses that outperform that of the best scoring ResNet-34. However, given that 89 experiments were performed with ResNet-18-FT, compared to only 21 with ResNet-34-FT, we believe that training the ResNet-34 further, and performing a more fine hyperparameter search would likely yield performance characteristics competitive with those of the best ResNet-18 trials.

The SqueezeNet, which we trained from a pre-trained state only, did not appear to perform well compared to the other models. Furthermore, the Inception-ResNet-v2 failed to generalize, consistently achieving validation losses well above 1.0.

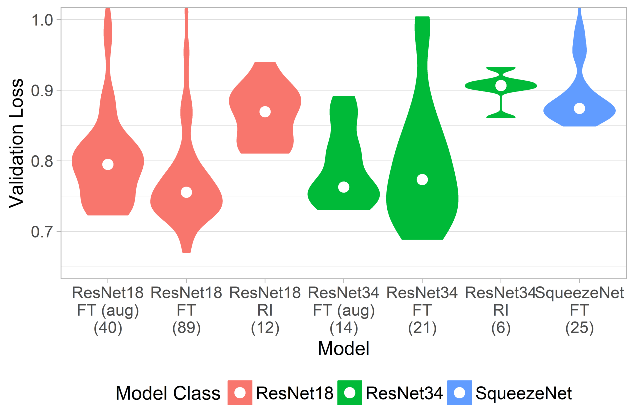


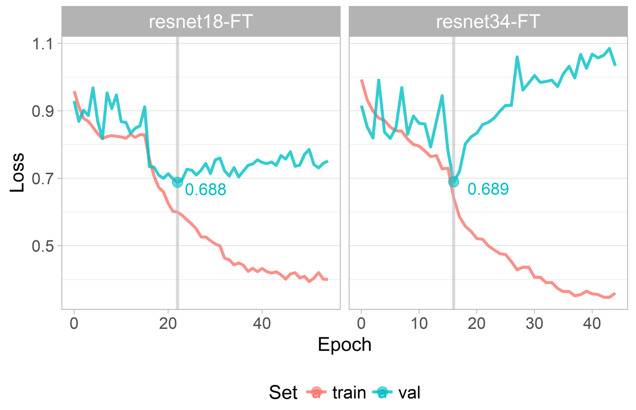
Figure 4: Distributions of the minimum validation loss achieved with each condition (architecture × initialization) over the course of our experiments. The white circle indicates the location of the mean of each distribution. The numbers in parentheses indicate the number of experiments performed with each condition. Abbreviations: FT: fine-tune, RI: random initialization.

Representative examples of the training-time loss dynamics of two high-performing ResNet models are given in Figure 5(a). In both cases, the model noisily attains a minimum of validation loss (ResNet-18-FT min. validation loss 0.688; ResNet-34-FT min. validation loss 0.689) early on in training (often around epoch 20), and as it begins to train, goes on to overfit, as evidenced by the training loss steadily approaching zero, and the validation loss plateauing, or even increasing, as in the case of the ResNet-34. It is interesting to note that, in both cases, the class-level F_1 scores (Figure 5(b)) on the validation set continue to improve, even as the models overfit the training set. We suspect that this is a result of overconfident incorrect predictions leading to increase in the validation cross-entropy loss without a significant change in the identity of the labels that are predicted. However, the observation that the validation loss of the ResNet-18 stabilizes, taken with its less noisy validation F_1 scores compared to those of the ResNet-34, suggest that using a shallower – and thus less complex – model affords a greater extent of training stability.

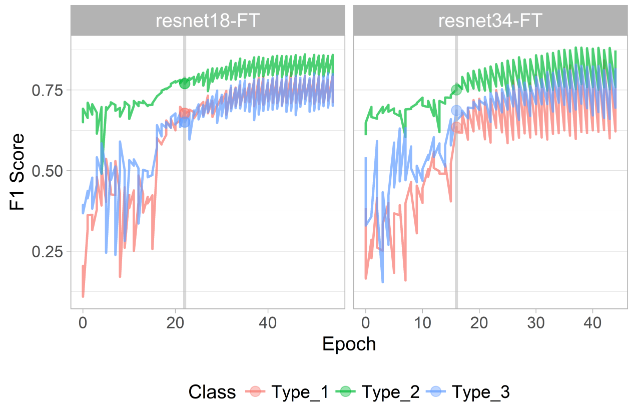
5.2. Performance on internal test set

5.2.1 Single models

For each model class, the parameters of the model that attained the minimal loss on the validation set were saved and used to generate predictions on the internal test set. The test-time F_1, accuracy, and loss for each of those models are shown in Table 3. These results demonstrate that the highest performing single model is the ResNet-34 with whole network fine-tuning, as it attains a test set loss of 0.70 and



(a) Training and validation loss dynamics for ResNet-18 and ResNet-34 models that underwent whole-network fine-tuning. These trajectories are representative of those models which attained high performance on the validation set.



(b) Class-level F_1 scores on the validation set for both models shown in (a).

Figure 5: Training and validation loss (a) and class level F_1 scores (b) for 18 and 34 layer ResNet models that underwent whole network fine tuning.

class-level F_1 scores of 0.71, 0.80, and 0.67 for Types 1, 2, and 3, respectively.

Overall, it is clear that fine-tuning an already pre-trained model is a more fruitful strategy compared to training from a random initialization, as both the ResNet-18 and ResNet-34 see a large boost in performance between the two initialization strategies (ResNet-18-RI min. test loss: 0.97 vs. 0.77 for FT, ResNet-34-RI min. test loss: 0.79 vs 0.70 for FT). However, it is interesting to note that training from a random initialization still appears to yield reasonable performance in the case of the ResNet-34, where it achieved a loss of 0.79, which compares well with the loss of 0.77 that was achieved by the fine-tuned ResNet-18.

5.2.2 Data Augmentation

Data augmentation was explored in both the context of creating an augmented set of training data and in the context of "data ensembling", as previously discussed in the methods. The results Table 3 indicate that the "data ensembling" approach is effective at providing a minor but consistent improvement the performance of single ResNet-18-FT and the ResNet-34-FT models, (ResNet-18-FT min. test loss: 0.77 vs. 0.72 for ResNet-18-FT†, ResNet-34-FT min. test loss: 0.70 vs 0.68 for ResNet-18-FT†). However, it is unclear whether the use of data augmentation at train time actually improves generalization, as performing these augmentations appeared to have the effect of improving the test set loss in the case of ResNet-18, while negatively impacting the both the loss and the class-level F_1 scores achieved by the ResNet-34.

5.2.3 Model Ensembles

Our results indicate that the Top-K mean ensembling strategy is highly effective at improving generalization of the models tested (Table 3), as our Top-5 Ensemble is able to achieve a test set loss of 0.58. There does not appear to exist one ensemble that consistently out-performs all other ensembles, but it is worth noting that each ensemble outperforms all individual models on all evaluation metrics. However, the diverse ensembling method proved less effective, as it appeared to perform worse compared to the Top-K approaches.

5.3. Performance on Kaggle leaderboard

As of June 8, 2017, our best-scoring submission to the Kaggle leaderboard had attained a Phase 1 public leaderboard position of 19th out of over 800 teams in the competition, with a multi-class cross-entropy loss on the Kaggle test set of 0.557. The performance of a selected subset of our models and ensembles on the Kaggle test set over time are summarized in Table 4. In general, our results on on the internal test set presented in Table 3 appear to correspond well to the Kaggle test results in Table 4. In particular, we find that the Top-K ensemble methods consistently out-performed submissions derived from a single model and that the Top-7 mean ensemble attains the best performance, although the Top-10 and Top-5 ensembles seem to be competitive. Unlike the evaluations on the internal test set, the "data ensembling" method did not appear to improve test time performance for the ResNet-34-FT model, as we achieved a Kaggle test set loss of approximately 0.65 both with and without the application of this method.

We hypothesize that the improvement in performance realized by ensembling our models derives from stabilizing overconfident predictions, particularly overly confident incorrect predictions, as those have a disproportionate effect

	F_1				
Model or Ens.	Type 1	Type 2	Type 3	Acc.	Loss
ResNet-18-RI	0.33	0.61	0.45	0.52	0.97
ResNet-18-FT	**0.72**	0.75	0.60	0.70	0.77
ResNet-34-RI	0.50	0.74	0.63	0.68	0.79
ResNet-34-FT	0.71	**0.80**	**0.67**	**0.74**	**0.70**
SqueezeNet-FT	0.48	0.71	0.48	0.62	0.91
ResNet-18-FT*	0.39	0.72	0.61	0.65	0.74
ResNet-34-FT*	0.50	0.65	**0.62**	0.62	0.77
ResNet-18-FT†	**0.65**	**0.75**	0.57	0.65	0.72
ResNet-34-FT†	0.57	0.73	0.60	**0.68**	**0.68**
Top-10 Ensemble	0.81	**0.85**	**0.75**	**0.81**	0.63
Top-7 Ensemble	**0.82**	0.83	0.73	0.80	0.59
Top-5 Ensemble	0.78	0.83	0.73	0.79	**0.58**
Diverse Ensemble	0.65	0.80	0.69	0.75	0.68

Table 3: Performance metrics for each model or ensemble on the internal test set. The best scores for single models and for ensembles are bolded. The asterisk (*) indicates that the model was trained using the data augmentations described in §4.2.2, and the dagger (†) indicates that the test-time predictions were made using the "data ensembling" approach described in §4.2.3.

on the test loss. To investigate this hypothesis more thoroughly, we pooled the predicted probabilities for each class for each example in the Kaggle test set and each model in the Top-10 ensemble and plotted the kernel density estimate of the distribution of the probabilities for each class (Figure 6) with the distribution of the class probabilities for the mean ensemble overlaid. Before ensembling, the distributions of the predictions of individual models appear bimodal, having modes at the extremes of the interval [0, 1], and that ensembling spreads out the mass of these distributions, thus demonstrating that our ensembles are indeed performing regularization of the predicted probabilities of each class.

6. Conclusion and future work

We have described the development and implementation of a convolutional neural network-based algorithm for cervical TZ type classification from cervigram images. Our work demonstrates the viability of transfer learning approaches to this problem, as well as the applicability of ensemble methods for improving the generalization performance of our algorithm. We were able to progressively improve the performance of our models through extensive experimentation, which included testing various data augmentation strategies and data splitting approaches.

Owing to the success of our ensemble approach, it is

Network	Kaggle test set loss
Random guessing benchmark	1.00225
ResNet-34-FT	0.64971
ResNet-34-FT†	0.65466
Top-10 Ensemble	0.57671
Top-7 Ensemble	**0.55727**
Top-5 Ensemble	0.56271
Diverse Ensemble	0.64745

Table 4: Progression of our algorithm's results in the Kaggle competition. The random guessing benchmark, $-\frac{1}{N}\sum_i n_i \log \frac{n_i}{N}$, takes into account the distribution of each class i in the train data. The dagger (†) indicates that the model was trained using "data ensembling". Abbreviations: RI: random initialization; FT: fine-tuning.

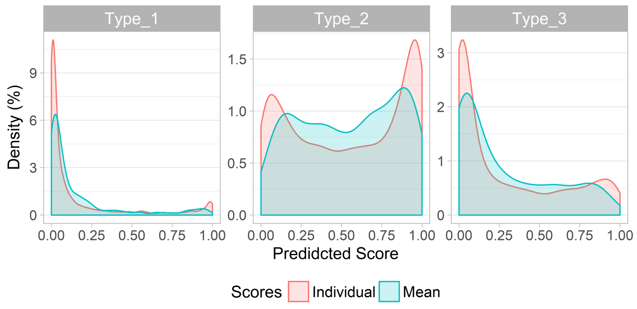


Figure 6: Distributions of predicted probabilities of each class for all images in the Kaggle test set, for the individual models in the Top-10 ensemble (red), compared to the distributions of the probabilities of the mean ensemble (blue).

likely that our algorithm could be improved by a more thorough exploration of methods capable of penalizing overconfident predictions, such as label smoothing [51] or maximum-entropy based confidence penalties [52]. Additionally, it may be worthwhile to explore data cleaning methods to systematically identify, for example, potential duplicate images for removal from the dataset in order to lessen potential biases introduced by their presence.

As this work was performed as a part of a currently ongoing Kaggle competition, our future work, at least in the short term, will focus on preparation of a final submission in the second phase of the competition that begins on June 15, 2017 and ends on June 21.

References

[1] B. Stewart and C. Wild, editors. *World Cancer Report 2014*. World Health Organization.

[2] W. Prendville. The treatment of cervical intraepithelial neo-

plasia: what are the risks? *Cytopathology*, 20(3):145–153, 2009.

[3] Kaggle. Intel & MobileODT Cervical Cancer Screening: Which cancer treatment will be most effective? URL https://www.kaggle.com/c/intel-mobileodt-cervical-cancer-screening.

[4] A. Razavian, H. Azizpour, J. Sullivan, and S. Carlsson. CNN Features Off-the-Shelf: An Astounding Baseline for Recognition. *CVPR*, 2014.

[5] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell. DeCAF: A Deep Convolutional Activation Feature for Generic Visual Recognition. *ICML*, 2014.

[6] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson. How transferable are features in deep neural networks? *NIPS*, 2014.

[7] H. Shin, H. Roth, M. Gao, L. Lu, Z. Xu, I. Nogues, J. Yao, D. Mollura, and R. Summers. Deep Convolutional Neural Networks for Computer-Aided Detection: CNN Architectures, Dataset Characteristics and Transfer Learning. *IEEE Transactions on Medical Imaging*, 35(5):1285–1298, 2016.

[8] K. He, X. Zhang, S. Ren, and J. Sun. Deep Residual Learning for Image Recognition. *arXiv: 1512.03385*, 2015. URL https://arxiv.org/abs/1512.03385.

[9] J. Deng, W. Dong, R. Socher, L. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. *CVPR*, 2009.

[10] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86:2278–2324, 1998.

[11] V. Gulshan, L. Peng, M. Coram, M. Stumpe, D. Wu, A. Narayanaswamy, S. Venugopalan, K. Widner, T. Madams, J. Cuadros, R. Kim, R. Raman, P. Nelson, J. Mega, and D. Webster. Development and Validation of a Deep Learning Algorithm for Detection of Diabetic Retinopathy in Retinal Fundus Photographs. *Journal of the American Medical Association*, 316(22):2402–2410, 2016.

[12] Y. Liu, K. Gadepalli, M. Norouzi, G. Dahl, T. Kohlberger, A. Boyko, S. Venugopalan, A. Timofeev, P. Nelson, G. Corrado, J. Hipp, L. Peng, and M. Stumpe. Detecting cancer metastases on gigapixel pathology images. *arXiv:1703.02442*, 2017.

[13] A. Esteva, B. Kuprel, R. Novoa, J. Ko, S. Swetter, H. Blau, and S. Thrun. Dermatologist-level classification of skin cancer with deep neural networks. *Nature*, 542:115–118, 2017.

[14] S. Lo, S. Lou, J. Lin, M. Freedman, M. Chien, and S. Mun. Artificial convolution neural network techniques and applications for lung nodule detection. *IEEE Transactions on Medical Imaging*, 14:711–718, 1995.

[15] Y. Zheng, D. Liu, B. Georgescu, H. Nguyen, and D. Comaniciu. 3D deep learning for efficient and robust landmark detection in volumetric data. *MICCAI*, 2015.

[16] D. Ciresan, A. Giusti, L. Gambardella, and J. Schmidhuber. Mitosis Detection in Breast Cancer Histology Images with Deep Neural Networks. *MICCAI*, 2013.

[17] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. *MICCAI*, 2015.

[18] K. Kamnitsas, C. Ledig, V. Newcombe, J. Simpson, A. Kane, D. Menon, D. Rueckert, and B. Glocker. Efficient multi-scale 3D CNN with fully connected CRF for accurate brain lesion segmentation. *Medical Image Analysis*, 36:61–78, 2017.

[19] M. Drozdzal, E. Vorontsov, G. Chartrand, and S. Kadoury. The importance of skip connections in biomedical image segmentation. *Lecture Notes on Computer Science: Deep Learning for Medical Image Analysis*, 10008:179–187, 2016.

[20] M. Simonovsky, B. Gutierrez-Becker, D. Mateus, N. Navab, and N Komodakis. A deep metric for multimodal registration. *MICCAI*, 2016.

[21] Y. Anavi, I. Kogan, E. Gelbart, O. Geva, and H. Greenspan. Visualizing and enhancing a deep learning framework using patients age and gender for chest X-ray image retrieval. *Medical Imaging. Vol. 9785 of Proceedings of the SPIE.*, 2016.

[22] W. Yang, Y. Chen, Y. Liu, L. Zhong, G. Qin, Z. Lu, Q Feng, and W. Chen. Cascade of multi-scale convolutional neural networks for bone suppression of chest radiographs in gradient domain. *Medical Image Analysis*, 35:421–433, 2016.

[23] J. Antony, K. McGuinness, N. Connor, and K. Moran. Quantifying radiographic knee osteoarthritis severity using deep convolutional neural networks. *arXiv:1609.02469*, 2016.

[24] E. Kim, M. Cortre-Real, and Z. Baloch. A deep semantic mobile application for thyroid cytopathology. *Medical Imaging. Vol. 9789 of Proceedings of the SPIE*, 2016.

[25] C. Szegedy, Wei Liu, Yangqing Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. *CVPR*, 2015.

[26] L. Mango. Computer-assisted cervical cancer screening using neural networks. *Cancer Letters*, 77(2-3):155–162, 1994.

[27] E. Kim and X. Huang. A data-driven approach to cervigram image analysis and classification. *Lecture Notes in Computational Vision and Biomechanics: Color Medical Image Analysis*, 6:1–13, 2013.

[28] D. Song, E. Kim, and X. Huang. Multi-modal entity coreference for cervical dysplasia diagnosis. *IEEE Transactions on Medical Imaging*, 34(1):229–245, 2015.

[29] T. Xu, H. Zhang, X. Huang, S. Zhang, and D. Metaxas. Multimodal deep learning for cervical dysplasia diagnosis. *MICCAI*, 9901:115–123, 2016.

[30] T. Xu, H. Zhang, C. Xin, E. Kim, L. Long, Z. Xue, S. Antani, and X. Huang. Multi-feature based benchmark for cervical dysplasia classification evaluation. *Pattern Recognition*, 63: 468–475, 2017.

[31] A. Khrizhevsky, A. Sutskever, and G. Hinton. ImageNet classification with deep convolutional neural networks. *NIPS*, 2012.

[32] L. van der Maaten and G. Hinton. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9:2579–2605, 2008.

[33] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. *arXiv preprint arXiv:1603.05027*, 2016.

[34] A. Veit, M. Wilber, and S. Belongie. Residual Networks Behave Like Ensembles of Relatively Shallow Networks. *arXiv:1605.06431*, 2016.

[35] A. Paszke, S. Gross, and S. Chintala. PyTorch: Tensors and Dynamic neural networks in Python with strong GPU acceleration. *http://pytorch.org*.

[36] S. Chilamkmurthy. PyTorch: Transfer Learning Tutorial. URL http://pytorch.org/tutorials/beginner/transfer_learning_tutorial.html.

[37] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *ICML*, 2015.

[38] X. Glorot, A. Bordes, and Y. Bengio. Deep Sparse Rectifier Neural Networks. *AISTATS*, 2011.

[39] R. Hahnloser, R. Sarpeshkar, M. Mahowald, R. Douglas, and H. Seung. Digital selection and analogue amplification coexist in a cortex-inspired silicon circuit. *Nature*, 405:947–951, 2000.

[40] N. Cullen. Torchsample: High-Level Training, Data Augmentation, and Utilities for PyTorch. URL https://github.com/ncullen93/torchsample.

[41] J. Bergstra and Y. Bengio. Random Search for Hyper-Parameter Optimization. *Journal of Machine Learning Research*, 13:281–305, 2012.

[42] Forrest N. Iandola, Song Han, Matthew W. Moskewicz, Khalid Ashraf, William J. Dally, and Kurt Keutzer. SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5MB model size. *arXiv:1602.07360*, 2016.

[43] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. Alemi. Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning. *AAAI*, 2017.

[44] D. Kingma and J. Ba. Adam: A Method for Stochastic Optimization. *ICLR*, 2015.

[45] Y. Nesterov. A method of solving a convex programming problem with convergence rate $\mathcal{O}(1/\sqrt{k})$. *Soviet Mathematics Doklady*, 28:372–376, 1983.

[46] I. Sutskever, J. Martens, G. Dahl, and G. Hinton. On the importance of initialization and momentum in deep learning. *ICML*, 2013.

[47] L. Breiman. Bagging predictors. *Machine Learning*, 24(2): 123–140, 1996.

[48] P. Sollich and A. Krogh. Learning with ensembles: How overfitting can be useful. *NIPS*, 1996.

[49] D. Maji, A. Santara, P Mitra, and D. Sheet. Ensemble of Deep Convolutional Neural Networks for Learning to Detect Retinal Vessels in Fundus Images. *arXiv:1603.04833*, 2016.

[50] A. Kumar, J. Kim, D. Lyndon, M. Fulham, and D. Feng. An Ensemble of Fine-Tuned Convolutional Neural Networks for Medical Image Classification. *IEEE Journal of Biomedical and Health Informatics*, 21(1):31–40, 2017.

[51] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the Inception Architecture for Computer Vision. *CVPR*, 2016.

[52] G. Pereyra, G. Tucker, J. Chorowski, L. Kaiser, and G. Hinton. Regularizing neural networks by penalizing overly confident output distributions. *ICLR*, 2017.