Deep Learning Approaches for Determining Optimal Cervical Cancer Treatment

Chaitanya Asawa
Stanford University
Dept. of Computer Science

Yushi Homma
Stanford University
Dept. of Mathematics

Stuart Sy
Stanford University
Dept. of Computer Science

Abstract

Cervical cancer can be effectively prevented if detected in the pre-cancerous stage. However, the treatment type is dependent on physiological differences, and often this requires medical expertise to determine. We use deep learning approaches in computer vision, such as Convolutional Neural Networks and Transfer Learning, to classify an image of a cervix as one of 3 different types. We experiment with training strategies such as Dropout, Batch Normalization, and Image Augmentation, and dataset strategies such as scaling the loss function and cropping. We also incorporate image preprocessing techniques, filtering a given noisy additional dataset based on distance from the clean dataset distribution and image sharpness with the 2D Laplacian kernel filter. Our best model consists of an ImageNet trained ResNet as a feature extractor combined with a custom deep Inception-like residual network. With the categorical cross entropy loss as the score metric, our best model is currently 122nd place out of 829 teams on the Kaggle leaderboard for this task with a loss of 0.7451.

1. Introduction

In 2012, there were an estimated 528,000 cases of cervical cancer worldwide, of which 266,000 were fatal [6]. Approximately 85% of those cases occurred in less developed regions. Today, there exist treatment methods that can effectively prevent cervical cancer if detected in its pre-cancerous stage. Even in low-resource settings, early stage cervical cancer can be identified and treated in a single visit. However, one of the greatest challenges of these treatment programs is determining the appropriate method of treatment. Different methods, depending on physiological differences, have varying effectiveness and often times in rural areas health care providers are not able to discern which treatment is most effective. If the wrong treatment is applied, one may falsely believe that the patient has been cured, when in fact they have not been – posing a significant health risk.

In particular, the difference between various cervix types is quite small, depending specifically on a region known as the "transformation zone." Figure 1 describes visually the main difference between different cervix types, showing the difficulty of the problem, and Figure 2 displays sample images.

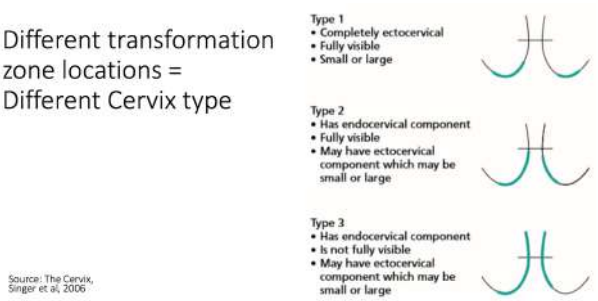

Figure 1: Differences between different type of cervices [9].

MobileODT and Intel have designed a Kaggle competition to tackle this particular issue and have released an associated dataset. In this paper, we attempt to develop an algorithm which, given an image of a cervix, uses Convolutional Neural Networks (CNNs) to output a womans cervix type. Specifically, we are investigating various CNN architectures, strategies such as Batch Normalization and Dropout, and as transfer learning to obtain the best possible classification.

Figure 2: Examples of different classes of cervices.

2. Related Work

To our knowledge, there is no published work on cervix type classification from images using deep learning because

currently, medical specialists perform this analysis by hand. The method that they use requires making an excision in the transformation zone and using a colposcopy to view it, and the specialist classifies the type based on the properties of the excision [2]. Of course, the weakness with this approach is that it requires a medical specialist.

Hence, we are motivated to use deep learning methods instead, based on their success in image classification - cutting edge models have even outperformed humans in the ImageNet challenge [7]. Examples of recently successful convolutional architectures are Inception-v3 [21] and ResNet [7], which we examine in detail in later sections. Previous work has also found techniques like Dropout [20], Batch Normalization [8], and Image Augmentation [11] to help with training and generalizing deep neural networks. Transfer learning has also found much success recently by using models pretrained on other tasks to train models on tasks that have much smaller amounts of data available [12, 1, 14, 13].

Additionally, deep learning approaches have found substantial success in medical imaging classification and detection tasks, such as dermatologist-level skin cancer classification [5], breast cancer growth characterization [18] and tumor tissue classification [23] among others [22, 15, 3]. For example, in performing dermatologist-level skin cancer classification, Esteva et. al apply transfer learning, using Google's Inception-v3 architecture pretrained on ImageNet, removing the final classification layer and adding their own classification layer. They then retrain the entire model, fine tuning all the layers, with their data. They were aided by curating a dataset consists of 129,450 clinical images, which is two orders of magnitude larger than previous datasets [5].

3. Methods

3.1. Convolutional Neural Networks

Convolutional Neural Networks (CNNs) are neural network architectures that employ convolution layers, which are layers that aim to capture meaning from spatial local connections. CNNs have demonstrated much success for image classification tasks, superior to previous methods, as demonstrated by Krizhevsky et. al [11] . For CNNs that do not involve pre-trained models, as described in the next section, we will distinguish them by referring to them to as *vanilla CNNs.*

3.2. Transfer Learning

Transfer Learning is a technique in which we use networks that have proven to do well on some task and try to adapt what is learned from this task to a separate but potentially related task. In our case, we would like to use networks that have performed very well at general image classification (specifically, on the ImageNet challenge, that tries to classify images into 1000 categories and has over one million training examples [16]), and try to adapt them to classifying images of cervices into different types.

4. Dataset

4.1. Description

We use the dataset provided by the aforementioned Kaggle competition [9]. The initial dataset the competition provides consists of the following distribution of classes:

Type	Number of Images	Percentage
I	250	17%
II	781	53%
III	450	30%

The competition also later provided additional data, however it is noted that some of these additional images might look alike they are taken in the same session, and some images were ones that were not selected because of image quality. In addition, sometimes there are other artifacts in these images such as gloves and some of the images are not even of a cervix. The amount and distribution of the additional data is as follows:

Type	Number of Images	Percentage
I	1191	18%
II	3567	53%
III	1976	29%

While the additional data significantly boosts the amount of data we have, because of the potential concerns with this data, we initially focused on using the initial data supplied and experimenting with image augmentation as described in section 5.2. We later experimented with methods of cleaning the additional dataset in order to include it in our training dataset.

4.2. Evaluation

Both for our loss function and for evaluating our model, we will use the categorical cross entropy loss, which is defined as

$$-\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{C} y_{ij} \ln \hat{y}_{ij}$$

where N is the number of examples, C is the number of classes, y_{ij} is 1 if the the ith example is truly of class j and 0 otherwise, and $\hat{y_{ij}}$ is the predicted probability that the ith example belongs to class j.

In practice, note for a given image the class we would be predicting is the one with the highest predicted probability. Our categorical accuracy is then

$$\frac{1}{N}\sum_{i=1}^{N} \mathbb{1}[\text{ argmax }_{j\in\{1,2,\ldots,C\}}\hat{y}_{ij} == \text{ argmax }_{k\in\{1,2,\ldots,C\}} y_{ik}]$$

However, we will focus on categorical cross entropy because it is better able to capture the strength of our predictions, and is also the metric the competition will be using for evaluation.

If we were to just predict classes with equal probability, we would expect that our loss in expectation would be -ln $(\frac{1}{3}) \approx 1.099$. However, as our data is not evenly distributed, if we predicted classes with probability proportional to the distribution, our loss in expectation is

$$-0.17 \ln (0.17) - 0.53 \ln (0.53) - 0.3 \ln (0.3) \approx 0.999$$

5. Technical Approach Details

We are using Keras as the main deep learning library with a Tensorflow backend [4].

5.1. Preprocessing

The preprocessing we do depends on our model's architecture, particularly because different base models we use for transfer learning may have different preprocessing approaches.

Resizing: Our images were initially of size 4096 by 3072 pixels. To reduce the number of parameters and subsequent computational time, we resize these images to 320 by 240 pixels.

Vanilla CNNs: With vanilla CNNs, we calculate the mean image across the training set, and subtract this from all the training examples, shifting the mean to 0. Note that we also shift the validation and test sets using the training set mean image.

Inception-v3, ResNet: For both Inception-v3 and ResNet, as they were trained on ImageNet, we subtract the mean image from the training set of ImageNet.

5.2. Strategies to Aid With Generalization

Dropout: Using dropout, while training, we randomly drop neurons and their connections with some probability p (a hyperparameter), aiming to prevent the neurons from co-adapting too much. At test time, we essentially average over all of the subsampled neural networks during training [20].

Batch Normalization: Batch Normalization is a recent technique that helps with the problem of properly initializing neural networks by explicitly forcing the activations throughout a network to take on a unit Gaussian distribution at the beginning of the training. Networks that use Batch Normalization are significantly more robust to bad initialization. Additionally, batch normalization can be interpreted as doing preprocessing at every layer of the network, but integrated into the network itself in a differentiable manner [8].

Image Augmentation: In addition to our initial data, we also generate more data using flips, translations, shears, and zooms, inspired by successful use of image augmentation by Krizhevsky et. al in the 2012 ImageNet Challenge [11]. These augmentations allow us to make our model invariant to transformations such as flips and translations, since they should be classified the same.

5.3. Transfer Learning Base Architectures

We have tried to do transfer learning with the following models, varying at which layer we get features from the different models and the architectures we specify on top of the outputs of these models:

Inception-v3: Inception-v3 aims to scale networks, specifically trying to make the additional computation as efficient as possible using inception modules. The goal of the inception module is to act as a "multi-level feature extractor" by computing 1×1, 3×3, and 5×5 convolutions within the same module of the network. The output of these filters are then stacked along the channel dimension and before being fed into the next layer in the network [21].

ResNet-50: First introduced by He et al. in their 2016 paper, Deep Residual Learning for Image Recognition [7], the ResNet architecture has become a seminal work, demonstrating that extremely deep networks can be trained using standard SGD (and a reasonable initialization function) through the use of residual modules which allow for better gradient flow through the network [7].

6. Baseline Results

As an initial baseline, we used a vanilla CNN with 14 sets of convolutions and 3 affine layers on the raw training data, excluding additional data and image augmentation. We also try some naive transfer learning by using Inception-v3 and ResNet as feature extractors and adding just a few convolution and affine layers on top of them.

One early result we find is that we are able to overfit to the initial training set fairly easily. With the ResNet model for example, we achieve 95% training accuracy, but validation accuracy of 44.93%, and this large gap leads us to indicate it is not doing any real learning and just memorizing the relatively small training set.

We summarize in the following table the best results, after tuning, we have been able to achieve with the following architectures on a self-designated validation set.

Model Type	Validation Loss	Validation Accuracy
Vanilla CNN	0.9723	51.52%
ResNet	1.0963	59.12%
Inception-V3	0.8953	60.47%

7. Inception-like Residual Network

We found that adding one or two layers to ImageNet models was not successful, and the model had difficulty learning. Hence, we decided to experiment with using the ImageNet models as a general feature extractor and then having our own more complex architecture take these extracted features and determine how to use them best for our task.

Specifically, we chose to use Inception-like layers with residual connections inspired by ResNet for our custom architecture. We chose such an architecture because residual connections would allow us to train deeper models, and Inception-like modules significantly reduces the number of parameters in the network, enabling more efficient training.

We detail our model below, using ResNet as a feature extractor:

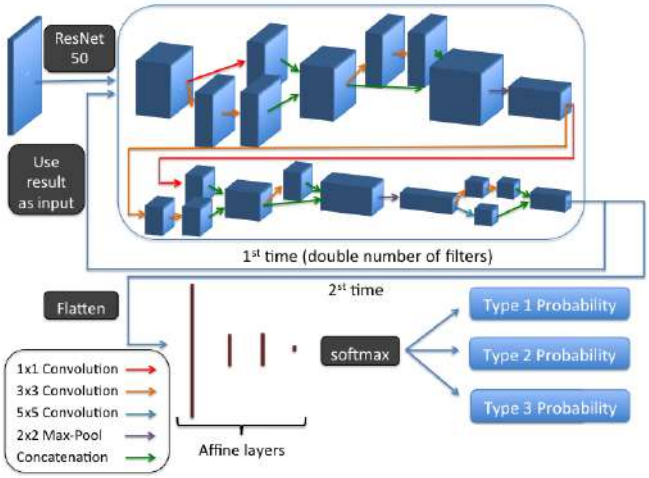

Figure 3: Architecture of our best model involving Inception-v3.

We find that this model does overfit, and so we use a dropout probability of 0.4 and L2 regularization of 5e-4 when experimenting with a model. We use the Adam optimizer and a learning rate decay of 0.01 each epoch.

8. Cropping

Looking more carefully at our data, we found that the transformation zone, which is the key to the classification decision in our problem, would often be in the center of the image. Additionally, in many of the images, there would be visual artifacts such as black borders or metallic speculums on the edges. Given this information, we decided to experiment with cropping out the center of the images so that the irrelevant visual artifacts could be removed. We tried cropping at several magnifications of the images, and the results of the experiment are below:

% Centered	Validation Loss	Validation Accuracy
100	0.8168	63.18%
85	0.8114	64.53%
70	0.7725	68.24%
55	0.8249	61.15%
40	0.8567	60.14%

A small amount of cropping did yield slightly better results, but overall this technique was too volatile to be used consistently. It is likely that the convolutional structure of our neural network would have allowed for our model to learn that those visual artifacts were irrelevant by itself, and we show this in Section 13.

9. Class Weights and Scaling the Loss Function

As noted in section 4, our dataset is quite unevenly distributed, and hence, it is very easy for our model to bias towards the more frequent classes.

We show a confusion matrix of one of our experiments using a model very similar to our best one:

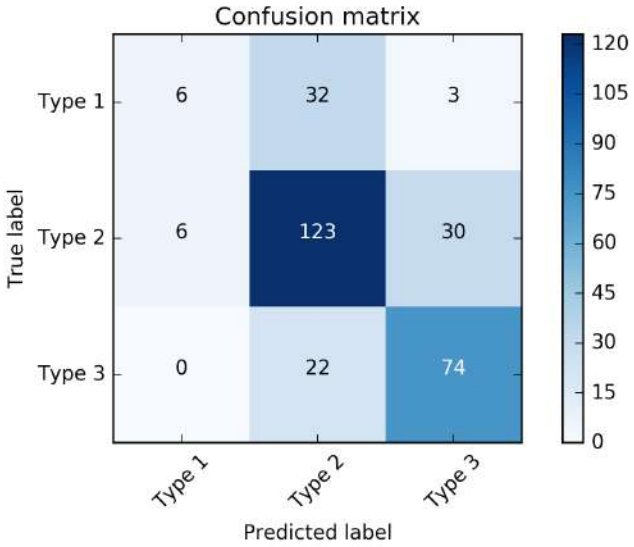

Figure 4: Confusion matrix

We find that our model is best at predicting class 2, the most frequent class, and so much so that it predicts most of the true class 1 images as class 2. For true class 3 images, we also predict a portion of the images to be class 2, but for

the most part we predict them correctly. We wanted to see if we could increase the recall in predicting class 1 and class 3, while not sacrificing much precision in predicting class 2. Hence, we experimented with penalizing our model differently for different classes. For classes which we weight more, in order to minimize the loss, the model tries to avoid making mistakes on an example of those classes more so than an example of another class.

Our modified loss function during training is then

$$-\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{C}\alpha_j y_{ij}\ln\hat{y}_{ij}$$

where α_j represents a scalar weight for class j.

We initially explored using weighting inversely proportional to the dataset distribution, which corresponds to weights of 5.88, 1.89, 3.33 for class 1, class 2, and class 3 respectively. We hoped this would need the model to be good across all classes, and not just class 2. Below, we show the results of some experiments with class weights.

Weights (1, 2, 3)	Validation Loss	Validation Accuracy
5.88, 1.89, 3.33	0.9931	48.31%
4.88, 1.89, 3.33	0.9696	54.39%
4.88, 1.89, 2.33	0.9652	56.08%
2.5, 1.0, 1.5	0.8583	57.77%

We find generally that class weights made it harder for the model to learn, with greater confusion across classes, and so we chose to keep equal weights for most of our experiments.

10. Additional Data

In this dataset from Kaggle, we were provided an additional dataset of images whose quality is not guaranteed to be as high as the initial dataset images, and sometimes even consists of completely irrelevant images. In order to effectively use this additional dataset in our training, we utilized two different techniques to clean the additional dataset.

First, we want the additional dataset to have images that actually show the cervix and are relatively similar to the initial dataset, since the test dataset is likely to have the same higher quality as the initial dataset. Hence, we want the additional dataset images to come from a similar distribution to that of the initial dataset.

To do this, we produced a vector representation of each image. This representation is generated by creating a histogram for each of the three channels based on the frequency of RGB values in the image, using 17 bins. This characterizes the color distribution in the image. We also did the same with each subrectangle of a 3×3 grid of the image, using 15 bins. This makes the representation take into account positioning of objects in the image. These histograms are concatenated together to create the vector representations of the images.

Once the images are converted into this vector format, we calculate the mean image and an approximate covariance matrix of the initial dataset. We assume that these images are samples of a multivariable Gaussian distribution with the aforementioned mean and covariance, and compute the pdf values of the additional dataset images. Each of these values is a relative similarity score of each additional image to the initial dataset images. Thus, we take only the additional images that are closest to the initial dataset by this metric.

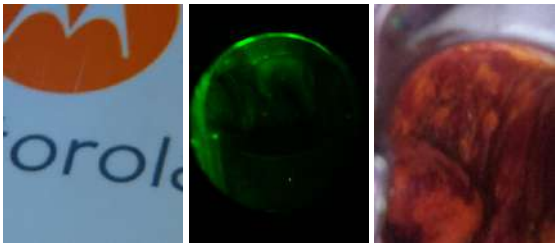

Figure 5: The three images above are examples of images in the additional dataset that were filtered out for being too different from the initial dataset.

This first technique looks only at the color distribution, but does not take into account the quality of the images. We use the sharpness of each image to characterize the quality of the image. To calculate sharpness, we convolve the center subrectangle of the images with the 2D Laplacian kernel filter and find the maximum value among all three channels and pixels. The idea is that we want the center of the image, which usually contains the relevant information, to be relatively sharp. We filter the additional dataset by including only images that at least as sharp as the least sharp image in the initial dataset. We then add this twice-filtered to the initial dataset to form our training dataset.

Figure 6: These three images are examples of images in the additional dataset that were filtered out due to blurriness (lack of sharpness).

In experiments involving the additional data, we limit the additional data to the training set, and have the validation

set come solely from the original dataset, hoping this would give us the closest measure of real performance on the test set. As we detail in coming experiments, we do find a gain of a couple hundredths in categorical cross entropy loss using the additional data.

We note though that, across training epochs with additional data, the score on the validation set is slightly unstable because of the differences that still do exist between training and validation data. We checkpoint our model at our best possible validation score to ultimately use for the competition.

11. Experiments With Feature Extraction

With the additional data, which we generally found helps our performance, we experimented with the feature extraction transfer layers, to see if with additional data we could better learn feature extraction.

11.1. Feature Extraction Model

We first compare the use (or not use) of different pre-trained models on ImageNet for feature extraction. In this case, the models are not being trained and we use the first 3 layers of the pre-trained models.

Feature Model	Validation Loss	Validation Accuracy
None	0.8016	63.51%
Inception-v3	0.7592	68.24%
ResNet	0.7157	69.93%

We find that pre-trained models for feature extraction do help as opposed to not having one, and we move forward with ResNet as a feature extractor.

11.2. Number of Layers of ResNet

We then explore how the number of layers of ResNet we use affect performance. The more layers of ResNet we use might result in extracting more and more ImageNet specific features, and so we try to find a good number of layers to use for feature extraction in our problem. Here, we train all of the layers so that, while we use more specific layers, we are able to hopefully adapt a little better to our dataset.

Num. Layers	Validation Loss	Validation Accuracy
3	0.7734	62.16%
6	0.7629	67.23%
12	0.7565	64.86%
20	0.7894	63.18%

We find, generally, very similar losses despite large changes to the number of layers. We use 6 layers then, because we find the higher accuracy is promising, as the model knows the correct class and can keep updating the probabilities to bring down the categorical crossentropy loss.

11.3. Training Layers of ResNet

Finally, we experiment with seeing the effects of fine-tuning layers of ResNet on performance.

Training	Validation Loss	Validation Accuracy
Yes	0.7734	62.16%
No	0.7157	69.93%

We find that not training layers actually helps performance – perhaps not training these layers keeps them more general rather than adapting more to the slightly different additional data.

12. Ensembling

Finally, once we optimized our custom architecture, our experiments with scaling the loss function with different class weights led to another idea: we wanted to try creating three separate models that slightly specialized more in predicting each cervix type (as they were penalized more for being incorrect for those classes), then ensembling the results of those models. We trained three models with cervix type 1, 2, and 3 weighted higher respectively, and generated predictions on the test set with each. Then, we combined them by using a weighted average for each probability value, weighted by the difference between 0.95 and the validation loss of that model. This weighting technique ensures higher contributions from the better scoring models. This resulted in our best test loss of 0.74517 on Kaggle, beating our model without ensembled class weightings.

13. Visualizing Our Model

To qualitatively evaluate our model, we applied several visualization techniques in order to gain an intuitive sense of what our model learned. We used *keras-vis* [10] to help generate many common types of neural network visualizations.

13.1. Saliency Maps

Saliency maps was first introduced in the paper: Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps [19]. The idea behind saliency maps is to determine which sections of the image are most important we compute the gradient of output category with respect to input image. This should tell us how output category value changes with respect to a small change in input image pixels. All the positive values in the gradients tell us that a small change to that pixel will increase the output value. Hence, visualizing these gradients, which are the same shape as the image, should provide some intuition of what our model is focused on when making classification decisions.

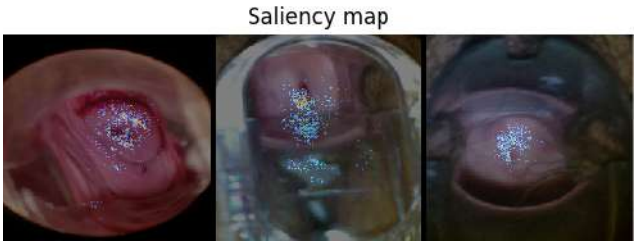

Figure 7: Saliency Maps for sample Type 1, 2 and 3 cervices.

The above image shows the saliency maps for three sample images of Type 1, Type 2, and Type 3 cervices taken from our training dataset. The dots show the locations in the image that contribute the most to the class prediction, and it appears that our model does focus on the transformation zone which is the actual physiological difference between the different cervix types.

13.2. Class Activation Maps

Similarly, we use class activation maps, proposed by Zhou et. al [24], which are an alternative way to visualize the discriminative image regions used by the CNN to identify the image label. We specifically the more robust grad-CAM method [17] to generate our visualizations.

Figure 8: Class Activation Maps for sample Type 1, 2 and 3 cervices.

The above image shows the class activation maps for three sample images of Type 1, Type 2, and Type 3 cervices taken from our training dataset. A brighter (redder) color of the overlay indicates that the region contributes more to the class prediction. Again, we can see that our model focuses mostly on the transformation zone of the cervix, and ignores background artifacts such as black borders and the metallic outlines of the speculums used in taking the images.

14. Conclusion

At the time of writing, our best model is 122nd place on the Kaggle leaderboard out of 829 teams. Overall, we found that a small amount of transfer learning combined with a powerful Inception-like residual network yielded the best results, as this was a fine discrimination task without much data. Ensembling several re-weighted versions of this model also increased test set performance.

In addition, we created our own technique to incorporate additional data, using only images close to our desired distribution and sharp enough as determined by the 2D Laplacian kernel filter, and we found this to help performance by extensive preprocessing and filtering of the dirty examples.

This was a difficult problem due to the minute visual differences between classes, and the relatively small biased training dataset, but we find that our model is able to pick up on these fine differences.

15. Future Work

There remains a lot to experiment with to continue improving our model. We found that adding a custom, complex architecture on top of a pre-trained model on ImageNet provided a substantial boost to our results, but there are many more architectures that we could experiment with.

Another experiment that could be interesting is, especially given the uneven distribution of the data, is create 3 separate models for each of the 3 classes that performs binary classification for that class. We could then bring together the results of those 3 models as our final prediction for the probabilities for each class our image belongs to. While such an approach is not feasible on tasks such as the ImageNet competition, which has a 1000 classes, it may be more reasonable in our scenario given that we only have 3 classes.

Finally, we also want to experiment with using different tolerances for the additional dataset cleaning. It is possible that we eliminated too much or too little of the additional dataset, so it would be beneficial to test what levels of similarity to the initial dataset and levels of sharpness to include in our training dataset.

References

[1] Y. Bengio et al. Deep learning of representations for unsupervised and transfer learning. *ICML Unsupervised and Transfer Learning*, 27:17–36, 2012.

[2] J. Bornstein, J. Bentley, P. Bösze, F. Girardi, H. Haefner, M. Menton, M. Perrotta, W. Prendiville, P. Russell, M. Sideri, et al. 2011 colposcopic terminology of the international federation for cervical pathology and colposcopy. *Obstetrics & Gynecology*, 120(1):166–172, 2012.

[3] T. Brosch, R. Tam, A. D. N. Initiative, et al. Manifold learning of brain mris by deep learning. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 633–640. Springer, 2013.

[4] F. Chollet et al. Keras. `https://github.com/fchollet/keras`, 2015.

[5] A. Esteva, B. Kuprel, R. A. Novoa, J. Ko, S. M. Swetter, H. M. Blau, and S. Thrun. Dermatologist-level classification of skin cancer with deep neural networks. *Nature*, 542(7639):115–118, 2017.

[6] GLOBOCAN. Cervical cancer estimated incidence, mortality and prevalence worldwide in 2012, 2012.

[7] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

[8] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[9] Kaggle. Intel and mobileodt cervical cancer screening, 2017.

[10] R. Kotikalapudi. keras-vis. `https://github.com/raghakot/keras-vis`, 2016.

[11] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[12] F.-F. Li. Knowledge transfer in learning to recognize visual object classes. *International Conference on Development and Learning*, 2006.

[13] S. Liao, Y. Gao, A. Oto, and D. Shen. Representation learning: a unified deep learning framework for automatic prostate mr segmentation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 254–261. Springer, 2013.

[14] G. Mesnil, Y. Dauphin, X. Glorot, S. Rifai, Y. Bengio, I. J. Goodfellow, E. Lavoie, X. Muller, G. Desjardins, D. Warde-Farley, et al. Unsupervised and transfer learning challenge: a deep learning approach. *ICML Unsupervised and Transfer Learning*, 27:97–110, 2012.

[15] A. Prasoon, K. Petersen, C. Igel, F. Lauze, E. Dam, and M. Nielsen. Deep feature learning for knee cartilage segmentation using a triplanar convolutional neural network. In *International conference on medical image computing and computer-assisted intervention*, pages 246–253. Springer, 2013.

[16] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.

[17] R. R. Selvaraju, A. Das, R. Vedantam, M. Cogswell, D. Parikh, and D. Batra. Grad-cam: Why did you say that? visual explanations from deep networks via gradient-based localization. *arXiv preprint arXiv:1610.02391*, 2016.

[18] M. Shah and C. Rubadue. Deep learning assessment of tumor proliferation in breast cancer histological images, 2016.

[19] K. Simonyan, A. Vedaldi, and A. Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*, 2013.

[20] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[21] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2818–2826, 2016.

[22] Y. Xu, T. Mo, Q. Feng, P. Zhong, M. Lai, I. Eric, and C. Chang. Deep learning of feature representation with multiple instance learning for medical image analysis. In *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*, pages 1626–1630. IEEE, 2014.

[23] X. Yang. A deep learning approach for tumor tissue image classification, 2016.

[24] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba. Learning deep features for discriminative localization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2921–2929, 2016.