# Ensemble Methods for Histopathology Classification

Rene Bidart
University of Waterloo
renebidart@gmail.com

## Abstract

*Evaluating metastasis of breast cancer throughout the body is important for deciding on future treatment plans for a patient. This involves the labour intensive process of pathologists manually checking tissue samples for cancer. As with many other image classification problems, convolutional neural networks haven been shown to be promising for this application. Because of the high-resolution of pathology images, current approaches use a tile level classifier to classify subsections of the raw image, generating a probability heatmap. Another classifier is applied to this heatmap to get the final classification. These two steps are separated, so increasing the accuracy of the classifier on the raw image will improve the final classification. Ensemble methods such as bagging and weighted ensembles have been shown to improve performance in many circumstances, so we investigate their effectiveness on this problem. We find that ensemble methods produce small improvements in performance, and also that bagging is relatively ineffective for ensemble models of CNNs*

*Code for this work is at* `https://github.com/renebidart/camelyon`

## 1. Introduction

To effectively treat breast cancer, doctors must evaluate the extent it has spread, or metastasized throughout the body. An important part of this is detecting metastasis in lymph nodes adjacent to the breast[2], which will be the focus of this work. To detect metastasis in lymph nodes, a pathologist must manually evaluate tissue samples. Unfortunately, this process is labour-intensive, error prone, expensive, and sufferers from inter-observer variability [16] [5].

Automated approaches have the potential to vastly improve this process, and convolutional Neural Networks (CNN) have recently seen good results on this problem[23][12][11]. Histopathology whole slide images (WSI) are extremely high resolution (≈10 billion pixels) in comparison to what is used in most networks today(≈100000 pixels). It is not possible to feed the entire image into the network at high resolution, so a different approach must be used than is common on an imagenet-style classification problem.

Currently, the most effective approaches to this problem take advantage of images that have pixel level annotations for the cancerous and non-cancerous regions, and convert this problem into one of classifying if a small region in the image contains cancer or not. A CNN is trained to classify these small regions in the image. By applying this trained classifier across the entire image, a heatmap of cancer probability is generated. Features are then extracted from this heatmap and fed into a standard classifier like SVM or gradient boosted trees to generate a slide-level prediction for the existence of cancer[23][12][11].

The CNN architectures used for this problem have been ones that performed well on imagenet, including GoogLeNet[20],VGG16[17][23] InceptionV3[21] [12] and ResNet[7][11]. Using transfer learning by pre-training models on imagenet before they are trained on the histopathology dataset did not improve the classification performance, but does significantly improve the training time of the model. These models were trained on patches between 224-299 pixels, similar to the size the original models were designed for. All approaches used dataset downsampling to deal with the abundance of normal tissue compared to cancerous tissue.

As an alternative to creating a dataset of patches to train the classifier on, some approaches[12][11] generated samples online during training. This means that random tiles were sampled for each batch, so there is some increase in the variety of samples seen by the network. The downside of this approach is that for each batch to be generated multiple WSIs must be read in, and because of their size this is a slow process. Because of time constraints we used the approach of generating a dataset initially rather than sampling online.

Current approaches to this problem divide it into two separate steps of first using a tile-level classifier to generate a heatmap and then classifying this heatmap, so we can separately improve these components to increase overall per-

formance. Improving this tile-level classifier will directly improve the the heatmap generated, and so will improve the performance of the overall classifier. In this paper we investigate improving the tile-level classifier using ensemble methods.

## 2. Data

The CAMELYON 2016 dataset [1] is used for this work, which is 400 whole slide images (WSI) divided into 270 for training and 160 for testing. Ground Truth is provided by a mask corresponding to each slide, which is an image with pixel level annotation indicating the cancerous regions. Both the mask and the WSI are very high resolution (100,000 x 200,000 pixels), with a single file being about 5 gigabytes. These are stored in a multi-resolution format, meaning that each file contains the high resolution image, as well as down sampled versions to a minimum size of about (512x1024). An example is shown in figure 1

Because the high resolution slides are too large to fit into the memory of a CNN, or even too large to perform simple operations on, openslide[6] is used to read in subsections of the WSI at a lower resolution.

### 2.1. Preprocessing

Because of the large size of the WSIs, we will segment out the background from the actual tissue to reduce the computation required. For this we use a preprocessing approach based off of[23]:

1. Read in WSI at resolution about 3072x7168 pixels, and convert from RGB to HSV
2. Use Otsu's algorithm[14] to separate the background from foreground, using the union of the result with the H and S channels, generating a tissue mask.

### 2.2. Dataset Generation

In many circumstances 270 images would be too few data points to train a CNN, but because we have pixel level annotation and high resolution images, there is effectively much more data. We create a dataset from subsections of the original image, using the labels from the pixel-level annotations, with a similar approach as in previous research.[22][23]

We create a dataset of 224x224 sized patches at the highest resolution because this is the most common input size for CNNs pre-trained on imagenet. Other research has also shown it is most useful to look at the WSIs at the highest level of resolution[23][12].

#### 2.2.1 Class Imbalance

The WSIs contain far more normal than tumor tissue, and because imbalanced data can be a problem for classifiers, we oversample the tumor class to create an dataset of half tumor and half normal patches.

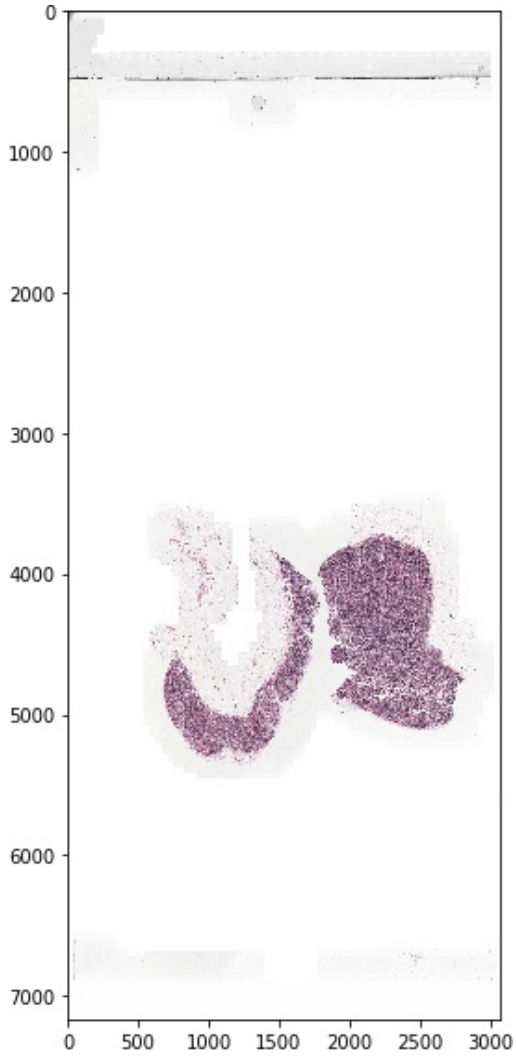


Figure 1. *Whole Slide Image (Non-cancerous), showing the large amount of irrelevant background in white.*

To generate the dataset, we alternate sampling tumor and non-tumor tissue. In the case of tumor, we select a tumor slide and sample from the region indicated by the tumor mask. For non-tumor, because cancerous slides also contain non-cancerous tissue, we select any slide, and make sure that the area we sample in is inside the tissue mask and but not in the tumor mask.

In both cases the tumor mask is down sampled to be the same resolution as the tissue mask, and sampling is done at this resolution. These points are then converted to the highest resolution, and we read in the tile at this level. No color normalization is used because it proved to be ineffective in other research [12].

## 3. Methods

### 3.1. Ensemble Classifiers

Given a set of classifiers, $\{h_1, h_2, ..., h_n\}$, an ensemble classifier is a weighted or unweighted average of the predictions of these classifiers. If the predictions are uncorrelated and better than chance, an ensemble classifier can perform better than any of the individual classifiers. [4] We can generate uncorrelated classifiers either by training different models, or by training the same models in a different way, such as on different datasets.

### 3.2. Bootstrap Aggregating

Bootstrap Aggregating, or bagging[3], is a way to generate ensemble classifiers by training the same model on different training sets. For each classifier, its training set is created by sampling the original dataset with replacement to create a new dataset the same size as the original dataset. This will mean that each classifier will see a different training set, and so will learn a different classifier. The effectiveness of this method will depend on how robust the models are to changes in their training data.

If models are stable with respect to changes in the training data, bagging will generate a set of similar models, and because ensembling requires uncorrelated models, the ensemble model will not increase performance significantly. On the other hand, if models are unstable, there is room for improvement with ensemble methods.

For this reason we think CNNs are a good candidate for bagging, because their high complexity should mean they will learn different functions based on differing data distributions. On the other hand, many regularization methods are used in CNNs such as dropout [18], weight decay, and early stopping of training, so they are much less prone to overfitting than their high number of parameters may suggest.

### 3.3. Models

We test a variety of networks on this dataset, including ResNet 18, 34, 50, 101 [7], VGG16 [17], ResNext 50, 101 [24] Inception-v4, Inception-ResNet-V2 [19] DenseNet 121, 169 [9].

These models are chosen because they have all been proven to perform well on the imagenet competition and have a very different architectures. Older models like VGG are basically stacks of convolutional and fully connected layers, ResNet introduced residual connections, and more modern architectures like DenseNet use dense connections between convolutional layers. The reason to test such a variety of architectures is because ensemble methods depend on models being uncorrelated, so because of the different architectures they should learn different classifiers.

### 3.4. Implementation

All models are implemented using PyTorch[15], also using the Fastai library[8]. All models were initialized with pre-trained weights on imagenet to improve the speed of convergence.

Training was done using the Adam optimizer[10], with the learning rate schedule defined by Stochastic Gradient Descent with Restarts [13]. Learning rates were defined differently for layers of the model, with the first third getting a learning rate of .001, the middle .005, and the last third .01. This is because the top level layers should learn features useful across all images, while the lower layers would have learned ones specific to imagenet. Augmentation was relatively standard for image classification, with random flips, rotations, and brightness transformations.

Bagging is implemented by randomly sampling from the dataset at the WSI level, rather than the patch level. We bootstrap sample from the list of WSIs, and then only select patches belonging to these WSIs. This is because sampling at the patch level will result in very similar samples because of the number of patches sampled, and so the models trained on this will be too correlated to find significant improvement through bagging.

Fine tuning only the final layer of the model gave 2% - 3% worse performance compared to retraining the full architecture across all models, so this method was not investigated further. In all results here the entire model was fine-tuned on the dataset.

## 4. Results

We attempted a variety of ensemble methods, from simple ensembles of the same model, to weighted ensembles of different models, as well as variants of bagging.

### 4.1. Simple Weighted and Unweighted Ensembles

Training the same model multiple times will result in slightly different classifiers because differences in the order the data is fed into the network will result in different weight updates. Unweighted models are created by averaging the predictions of all the CNNs, while the weighted classifiers will use a weighted average, weighting the predictions of the better models more highly.

Heterogeneous ensembles use multiple model architectures, so there is a much greater variance in the performance of the models. These are also combined using weighted or unweighted averages.

As shown in figure 2, there is an improvement in performance through using the unweighted ensemble method, reaching accuracy of 95.2%. This is a small improvement, and only about a .4% over the best performing single model, DenseNet 169. Weighted ensembles are not shown, because performance was within .1% of the unweighted model.

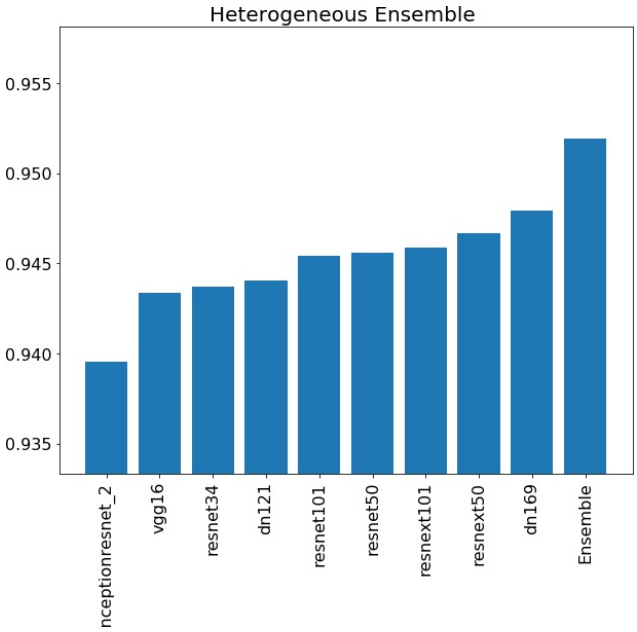


Figure 2. *Heterogeneous ensemble trained on the full dataset. The final column shows the accuracy of the unweighted ensemble of the predictions of all models, while the others show accuracy of each individual model*

Given the good performance of the DenseNet 169, we investigated ensembling this model only, using only the randomness induced by the training procedure to make these models uncorrelated. As can be seen in figure 3, the ensemble improved performance somewhat, but this was not significantly different than the heterogeneous ensemble, with accuracy at 95.1%.

#### 4.1.1 Model Performance and complexity

To more rigorously investigate how using weaker or more uncorrelated models affects performance in an ensemble, we trained ResNet 18, 34, 50, 101 on this problem. As shown in figure 4, looking at the average accuracy of 5 individual models compared to the ensemble accuracy indicates that for all models the ensemble improved performance. It is not clear that there is any relationship between the number of layers in the model and the effect of ensembling with ResNet.

### 4.2. Bagging

We also investigated bagging ensembles of one CNN architecture. This is created by training the same architecture on different bootstrapped samples of the dataset. This means that although the training set will be the same size as the original, it will contain duplicates and will on average only contain 63% of the samples in the original dataset. The final comparison of methods is shown in table 1. The

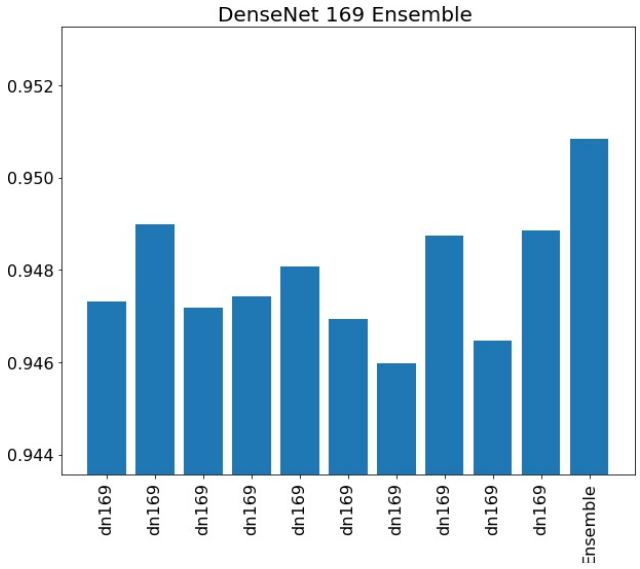


Figure 3. *DenseNet169 trained multiple times on the same dataset. The final column is the unweighted ensemble accuracy.*

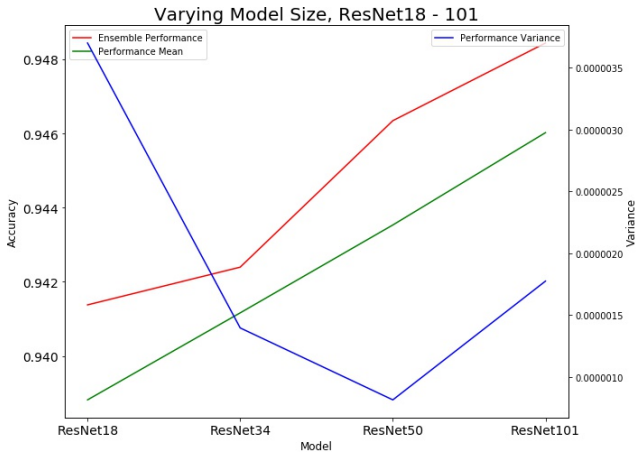


Figure 4. *Varying the number of layers in a ResNet model. The green line shows average performance of 5 models, while the red line shows the Ensemble performance. Ensemble models consistently improve performance regardless of model size.*

results indicate that bagging is an ineffective strategy, and the best results come from averaging well performing models trained on the full dataset. The individual models trained during bagging were significantly worse than those trained on the full dataset, and the final ensembling was not enough to make up for this poor performance.

### 4.3. Modified bagging

We also modify the bagging algorithm to sample without replacement, and adjust the number of samples taken. The reason is to directly investigate the effect of training more variable models, because the smaller dataset should produce

Table 1. Model Performance

| Model Description | Accuracy |
|---|---|
| Bagging ResNet50 (15 model) | 94.3% |
| Bagging DenseNet169 (15 model) | 94.5% |
| DenseNet169 Ensemble (10 model) | 95.1% |
| Heterogeneous Ensemble (9 model) | 95.2% |

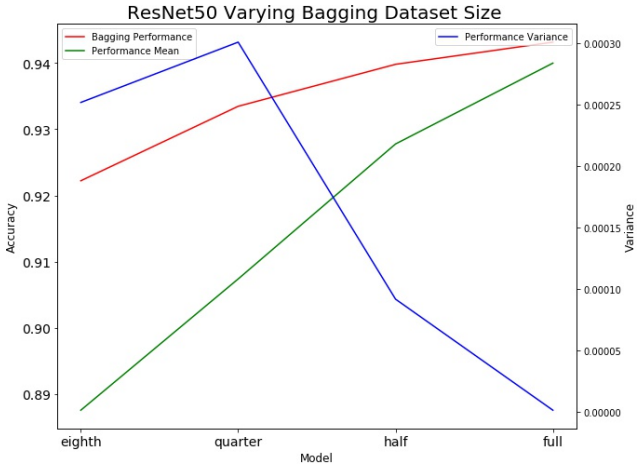


Figure 5. *Modified version of bagging, where ResNet50 is trained on part of the dataset, sampled without replacement. Indicates bagging gives greatest performance improvement when models are very uncorrelated, but the best strategy remains to train on the full dataset and average models normally.*

models that learn different classifiers.

We trained a standard model, ResNet50 on datasets sampled without replacement of size full, half, quarter and an eighth of the original dataset. 10 models were trained and averaged for each dataset size. As shown in figure 4, using a smaller dataset reduces the performance of individual models. Ensembling gives a larger improvement to these more variable models, as would be expected, but this is still not enough to make up for the performance gain of training models on the full dataset.

## 5. Conclusion

Bagging is an ineffective method for improving the performance of CNNs for histopathology classification. Bagging did not improve performance over simple model averaging. Both averaging the same architecture trained multiple times, as well as ensembles of different well performing architectures gave similar performance, which was a slight improvement over the best performing single model.

Bagging should be useful where models are over fitting, and although neural networks are extremely complex models, because of the training procedures used they do not exhibit the significant overfitting that may be seen in other models such as decision trees, and so do not show the same amount of performance gain. Because of this bagging is not effective for this problem, and ensemble models in general do not give significant performance gains.

## References


[1] Camelyon 2016. `https://camelyon16.grand-challenge.org/`. Accessed: 2018-02-01.

[2] S. K. Apple. Sentinel lymph node in breast cancer: Review article from a pathologist's point of view. *Journal of pathology and translational medicine*, 50 2:83–95, 2016.

[3] L. Breiman. Bagging predictors. *Mach. Learn.*, 24(2):123–140, Aug. 1996.

[4] T. G. Dietterich. Ensemble methods in machine learning. In *Multiple Classifier Systems*, pages 1–15, Berlin, Heidelberg, 2000. Springer Berlin Heidelberg.

[5] J. G. Elmore, G. M. Longton, P. A. Carney, B. M. Geller, T. Onega, A. N. Tosteson, H. D. Nelson, M. S. Pepe, K. H. Allison, S. J. Schnitt, et al. Diagnostic concordance among pathologists interpreting breast biopsy specimens. *Jama*, 313(11):1122–1132, 2015.

[6] A. Goode, B. Gilbert, J. Harkes, D. Jukic, and M. Satyanarayanan. OpenSlide: A vendor-neutral software foundation for digital pathology, 2013.

[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] J. Howard et al. fastai. `https://github.com/fastai/fastai`, 2018.

[9] G. Huang, Z. Liu, K. Q. Weinberger, and L. van der Maaten. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, volume 1, page 3, 2017.

[10] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[11] B. Lee and K. Paeng. Breast cancer stage classification in histopathology images. 2017.

[12] Y. Liu, K. Gadepalli, M. Norouzi, G. E. Dahl, T. Kohlberger, A. Boyko, S. Venugopalan, A. Timofeev, P. Q. Nelson, G. S. Corrado, et al. Detecting cancer metastases on gigapixel pathology images. *arXiv preprint arXiv:1703.02442*, 2017.

[13] I. Loshchilov and F. Hutter. Sgdr: stochastic gradient descent with restarts. *arXiv preprint arXiv:1608.03983*, 2016.

[14] N. Otsu. A threshold selection method from gray-level histograms. *IEEE Transactions on Systems, Man, and Cybernetics*, 9(1):62–66, Jan 1979.

[15] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer. Automatic differentiation in pytorch. 2017.

[16] S. S. Raab, D. M. Grzybicki, J. E. Janosky, R. J. Zarbo, F. A. Meier, C. Jensen, and S. J. Geyer. Clinical impact and frequency of anatomic pathology errors in cancer diagnoses. *Cancer*, 104(10):2205–2213, 2005.

[17] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

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

[19] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. In *AAAI*, volume 4, page 12, 2017.

[20] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. E. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. *CoRR*, abs/1409.4842, 2014.

[21] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the inception architecture for computer vision. *CoRR*, abs/1512.00567, 2015.

[22] Y. S. Vang, Z. Chen, and X. Xie. Deep learning framework for multi-class breast cancer histology image classification. *CoRR*, abs/1802.00931, 2018.

[23] D. Wang, A. Khosla, R. Gargeya, H. Irshad, and A. H. Beck. Deep learning for identifying metastatic breast cancer. *arXiv preprint arXiv:1606.05718*, 2016.

[24] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He. Aggregated residual transformations for deep neural networks. In *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on*, pages 5987–5995. IEEE, 2017.