Mole Classification: Malignant vs. Benign

Andrew Steffen Nathan Rice

Introduction

In this project, we work with the Kaggle dataset "Skin Cancer: Malignant vs. Benign"[1]. Our goal is to implement models that predict whether a mole is malignant (cancerous) or benign, given input which is an image of the mole.

The dataset is from the ISIC Archive (International Skin Imaging Collaboration). The input data are color images of size 224x224x3. The images were taken using a dermatoscope, a medical device that allows for inspection of skin lesions without obstruction by skin surface reflections. The data belong to one of two classes, "benign" or "malignant". The training set contains 1441 benign examples and 1198 malignant examples. The test set contains 361 benign examples and 301 malignant examples. The data are relatively balanced between the two classes.

The GitHub repository for our project is: https://github.com/asteffen/mole-classification

Contributions of each team member:

- Nathan wrote the section on VGG.
- Andrew wrote every other section.

Related Work

In this project, we use CNNs (convolutional neural networks) because they are an effective model for image classification. CNNs are neural networks that contain several types of layers, including convolutional layers and fully connected layers. In convolutional layers, the output is calculated by sliding a filter over the input image and computing the dot products.

[1] Skin cancer dataset: https://www.kaggle.com/fanconic/skin-cancer-malignant-vs-benign

Convolutional layers are locally connected layers with shared weights. CNNs preserve some spatial information from the image, because instead of flattening the image into 1 dimension, we keep the original 2-dimensional structure of the image.

Yann LeCun is one of the inventors of CNNs. In 1998, Yann LeCun et al. published a pioneering paper about a CNN that classifies handwritten digits which was applied at several banks to read checks[2]. Large-scale image classification has become possible due to large datasets such as ImageNet and fast GPUs. The ImageNet Large Scale Visual Recognition Challenge (ILSVRC) held a competition each year from 2010 to 2017, serving as a benchmark to judge state-of-the-art image classification systems. In 2012, Alex Krizhevsky et al. published a paper describing a CNN architecture called AlexNet which they used to win ILSVRC. The AlexNet paper was massively influential in the field of computer vision and motivated more research in CNNs. In 2014, Karen Simonyan and Andrew Zisserman created VGGNet, which was the runner-up in ILSVRC that year. In 2015, Kaiming He et al. achieved 1st place in ILSVRC with ResNet, which they described in the paper "Deep Residual Learning for Image Recognition".

Method

VGG

In the VGG paper, researchers made a thorough evaluation of deep networks[3]. They increased the depth of the architecture by adding more convolutional layers, and made this beneficial through only using very small 3x3 convolution filters for all layers.

---

[2] Y. Lecun, L. Bottou, Y. Bengio and P. Haffner, "Gradient-based learning applied to document recognition," in *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278-2324, Nov. 1998. doi: 10.1109/5.726791

[3] Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556.

The input to the network is a fixed-size 224x224 RGB image. There is preprocessing, but it is only subtracting the mean RGB value, computed on the training set, from each pixel. The convolution stride is fixed to 1 pixel which means that the spatial resolution is preserved after convolution.

The researchers explain how using n 3x3 filters gives an output of the same size as one (2n+1)x(2n+1) filter (stride = 1 for all filters). They then go on to explain why using more filters is beneficial. Firstly, using more filters increases the number of non-linear rectification layers, which makes the decision function more discriminative. Secondly, more filters actually decreases the number of parameters.

The VGG16 model contains 12 convolution layers split into 5 groups by max pooling layers, and all the convolution layers use 3x3 filters. At the end of the convolution layers, there is another max pooling layer followed by 3 fully connected layers of sizes 4096, 4096, and 1000, and the softmax layer of size 2 for our 2 classes.

Deep Residual Learning for Image Recognition (ResNet)

In the ResNet paper, the researchers noticed that when they tried adding more layers to a convolutional neural network, the network suffered from a degradation problem[4]. The problem is that adding more layers to a suitably deep model leads to higher training error, which shows that the degradation problem is not caused by overfitting. The researchers note that it is possible to construct a deeper model that achieves a training accuracy that is at least as high as the shallower model's accuracy. In this construction, the deeper model contains a copy of all the layers from the shallower model, and the remaining layers are identity mapping.

[4] He, Kaiming et al. “Deep Residual Learning for Image Recognition.” 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2016): n. pag. Crossref. Web.

To alleviate the degradation problem, the researchers introduce shortcut connections which simply perform identity mapping. They call the networks with shortcut connections "residual networks" or ResNet, as opposed to "plain" networks. The residual networks are easier to optimize than plain networks, and they gain accuracy from greatly increased depth. An intuitive explanation of why the shortcut connections are effective is that during each convolution layer, some of the spatial information from the image is lost. The shortcut connections improve accuracy by preserving spatial information from earlier layers.

The ResNet34 model contains 34 convolutional layers. All the convolutional layers use 3x3 filters, except for the first one which uses 7x7 filters. We use the default PyTorch implementation of ResNet34, which has some differences from the paper. For example, the PyTorch model contains 2 dropout layers, but the ResNet paper does not use dropout. At the end of the convolutional layers, there is an average pooling layer followed by a Flatten layer that has output size 1024. This is followed by a fully connected layer with output size 512, then another fully connected layer with output size 2 which represent the activations for the 2 classes (malignant, benign).

Transfer Learning

For both ResNet34 and VGG16, we use transfer learning which is a technique where a model trained on one task is re-purposed on a second related task. When we initialize the model, we start with weights that are pre-trained on ImageNet. Transfer learning works because when the original model is trained on ImageNet, it learns features which generalize to mole classification.

We train the model for a total of 200 epochs. During the first 40 epochs of training, all convolutional layers of the network are frozen and the 2 fully-connected layers at the end are unfrozen. A layer being "frozen" means its weights are not updated. Before the next 160 epochs

of training, we unfreeze the model so that all layers are trainable. The reason we use transfer learning is that our dataset is small. With only about 1500 images per class, we suspect that we could not achieve the same accuracy without transfer learning.

Experiment

Experimental Setup

In this project, we used fastai, which is a deep learning framework built on top of PyTorch. The computing platform we used is Google Cloud Platform. We used a cloud instance with 4 vCPUs, 15 GB RAM and one GPU (NVIDIA Tesla K80).

We defined the validation set by randomly splitting the training set with an 80/20 split. For data augmentation, we apply transformations to the images during training. The transformations we use are horizontal flipping and rotation (by multiples of 90 degrees). Data augmentation increases the effective size of the dataset by 8 times, and serves the purpose of reducing overfitting in our model.

We trained the model using the 1cycle policy, which is a policy for adjusting the learning rate and momentum within each epoch. In a 2018 paper, Leslie Smith described the 1cycle policy which reduces the number of epochs needed to train a model[5]. To help us decide the learning rate, we used the lr_find function from fastai. This function tries out a range of learning rates for 100 iterations of training, and generates a plot of learning rate vs. loss. While the model is frozen, we use a learning rate that ranges from 1×10^{-5} to 1×10^{-3}. After unfreezing the model, we use a learning rate that ranges from 1×10^{-6} to 1×10^{-4}. We found that these learning rates were suitable for both ResNet34 and VGG16.

[5] Smith, L.N. (2018). A disciplined approach to neural network hyper-parameters: Part 1 - learning rate, batch size, momentum, and weight decay. ArXiv, abs/1803.09820.

Experimental Results

Figure 1 shows the training loss and validation loss for each epoch for the ResNet34 model. Figure 2 shows the training accuracy and validation accuracy for each epoch. For both the training and validation sets, the loss and accuracy were calculated with the model in PyTorch "eval" mode which means Dropout layers were disabled. Figures 3 and 4 show the loss and accuracy plots for VGG16.

From inspecting the graphs for ResNet34, we can see that some overfitting occurred. After epoch 125, the validation loss increases slightly. Also, the training accuracy rises above 99%. The epoch with the best validation accuracy was epoch 122, with a validation accuracy of 94.1%.

Figure 1. Loss for each epoch (ResNet34)

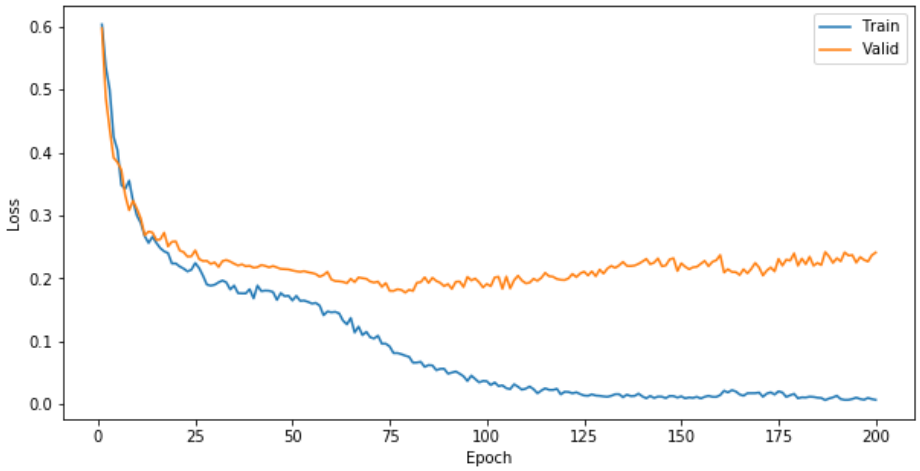


Figure 2. Accuracy for each epoch (ResNet34)

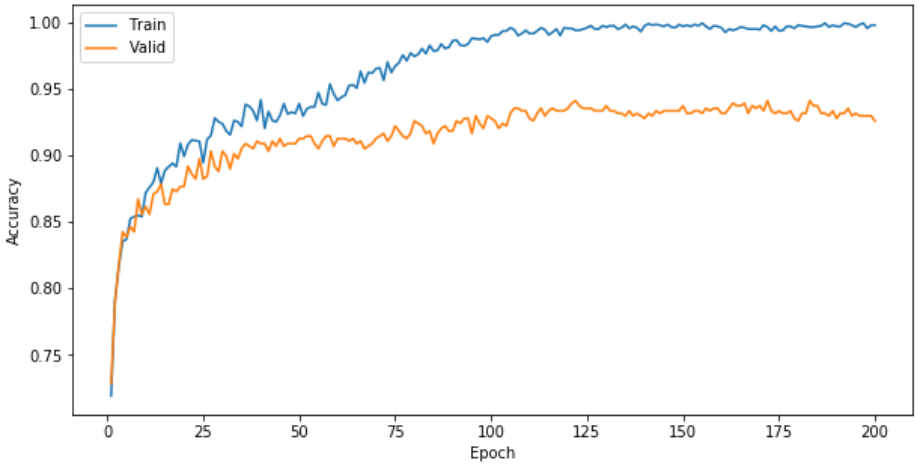


Figure 3. Loss for each epoch (VGG16)

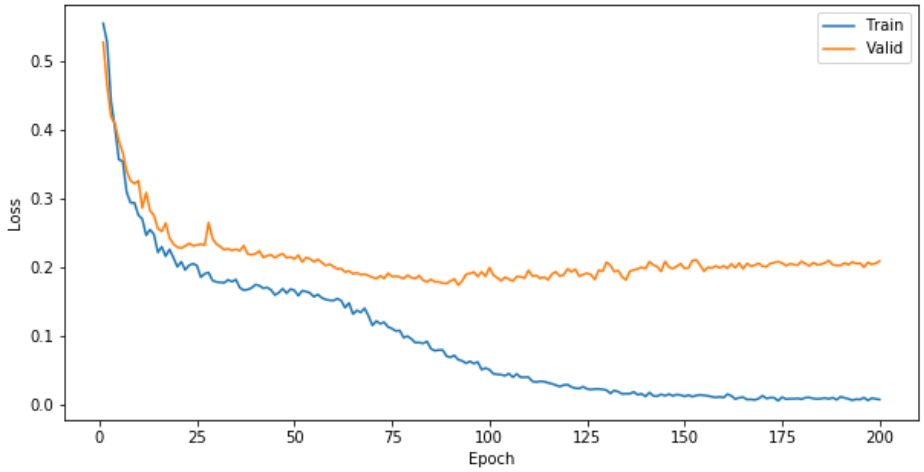
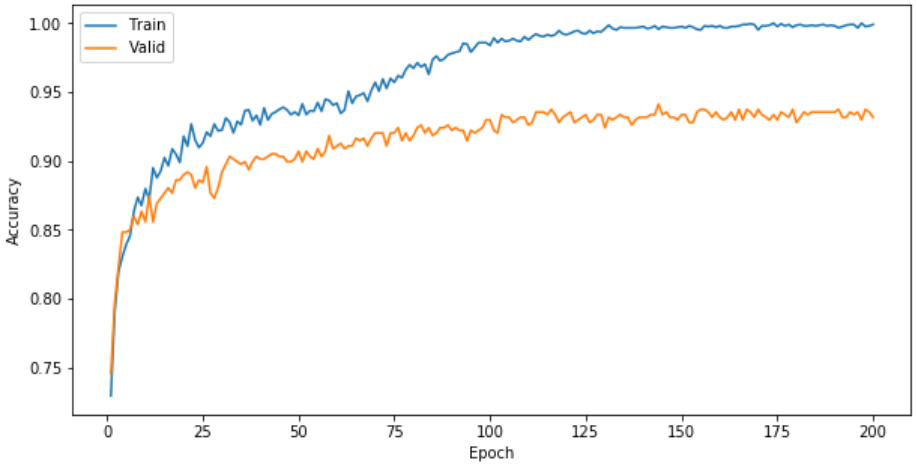


Figure 4. Accuracy for each epoch (VGG16)



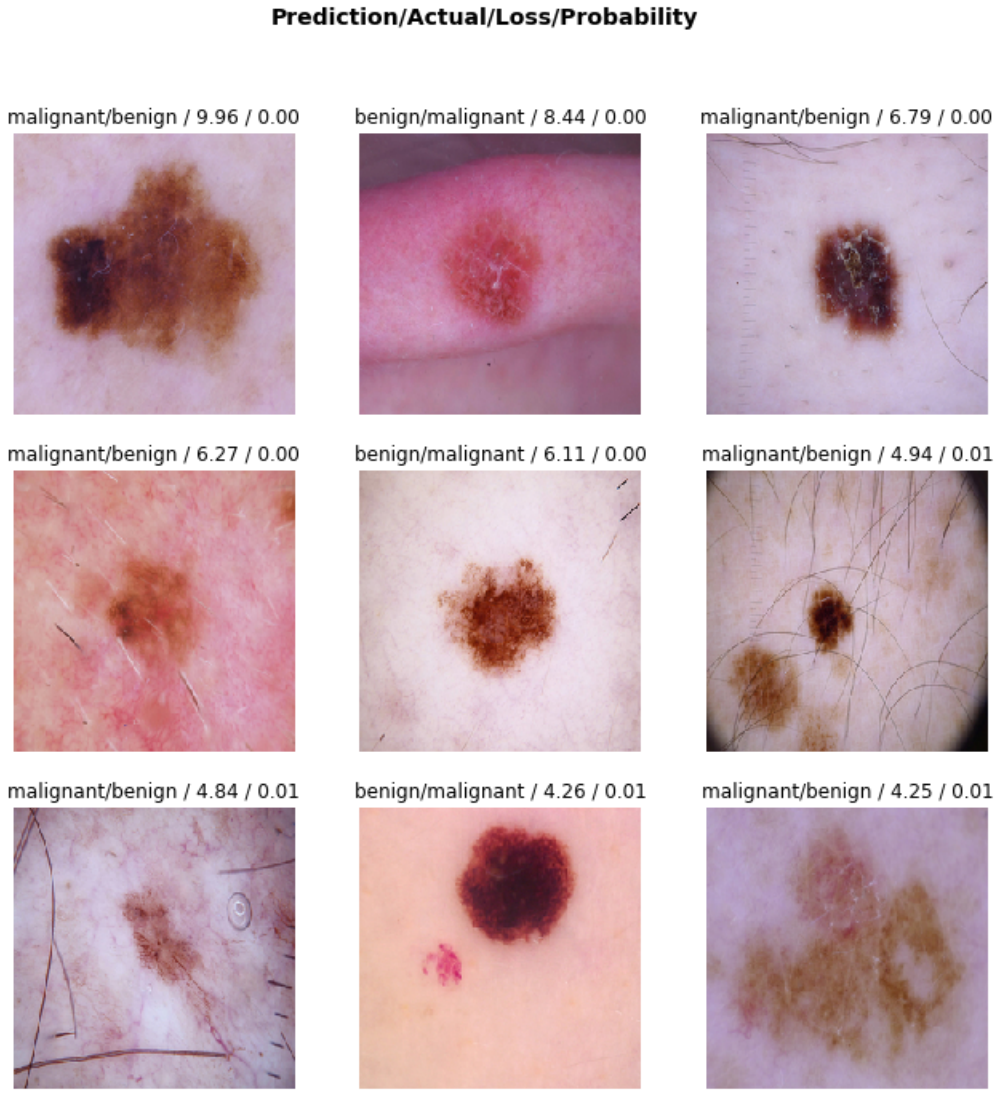
To evaluate the metrics for the test set, we loaded the best weights into the model (the weights with highest validation accuracy). The test set metrics for both models are shown in Table 1. VGG16 achieved a lower loss, and a slightly higher score for accuracy, precision, and recall.

Table 1. Test set metrics

	Loss	Accuracy	Precision	Recall	F1 score
VGG16	0.2920	0.9076	0.8893	0.9100	0.8995
ResNet34	0.3401	0.8970	0.8718	0.9067	0.8889

Figure 5 shows the 9 misclassified examples with the greatest losses (classified by ResNet34). Some signs that a mole is malignant are that it has an asymmetrical shape, its edge is not smooth, and it has uneven shading or dark spots. Because we do not have experience diagnosing moles, we are not sure why the model misclassified these examples.

Figure 5. The misclassified examples with the greatest losses (ResNet34)



Summary

We implemented VGG16 and ResNet34 models to predict whether a mole is malignant or benign, given as input an image of the mole. Both models were initialized with weights that were pre-trained on ImageNet. We found that VGG16 achieved a slightly higher accuracy on the test set compared to ResNet34. This is unexpected because in the ILSVRC, ResNet had much better accuracy than VGG. We are not sure of the reason that VGG achieved higher accuracy.

We think overfitting is not the reason, because we used the weights from the epoch with the best validation accuracy. For future work, a logical next step is to investigate why VGG16 out-performed ResNet34 using this dataset and the experimental setup we used.