CONVOLUTIONAL NEURAL NETWORKS FOR HISTOPATHOLOGY

Sheila Acar

MSDS 458 – Artificial Intelligence & Deep Learning

Tuesday, June 11, 2024

Abstract

Artificial intelligence is revolutionizing the field of medical diagnostics. In histopathology, the integration of artificial intelligence with digital image scanning is transforming the way that tissue samples are examined and diagnosed. The goal of this research study is to develop the most effective convolutional neural network for classifying different textures in colorectal cancer histology. Several architectural neural networks were developed using the colorectal_histology dataset in Tensorflow. First, the numbers of hidden nodes in the dense layer were tested. Next, the numbers of hidden layers were explored, with each layer featuring an increasing number of neural nodes. Once the number of hidden and the number of hidden layers were optimized, regularization techniques were added. These techniques include dropout, L2 regularization, batch normalization, and augmentation. Various dropout rates and L2 regularization rates were tested for optimization. In addition, the last two experiments explored two different optimizers, SGD and RMSprop. The best convolutional network developed features 3 hidden layers with 32, 64, and 128 filters, respectively; 128 units in the first dense layer; and dropout layers with a rate of 0.2. This model achieved a test accuracy of 77.8% and had closely matched training and validation accuracy values.

Introduction.

The potential for artificial intelligence to be used in developing tools that assist in diagnostics is promising. Such a tool can provide an extra layer of verification, ensuring that pathologists do not overlook critical details. In cancer diagnostics, early detection can be the difference between life and death. With the help of AI, cancer diagnostics can be more foolproof, ultimately saving more lives and improving patient outcomes. Pathologists use histological images to determine the timeline for follow-up colonoscopies, and AI can significantly enhance

this monitoring process. The goal of this research is to develop a tool that can classify histology images using the colorectal_histology dataset. The 8 different classes in the colorectal_histology dataset include tumor epithelium, simple stroma, complex stroma (stroma that contains tumor cells and/or single immune cells), immune cell conglomerates, debris and mucus, and mucosal glands, adipose tissue, and background. A successful neural network model that achieves high accuracy has the potential for real-world application as an assistive tool. A highly optimized diagnostic tool can help reduce diagnostic errors, speed up the diagnostic workflow, and provide consistency.

Literature review

Jakob Kather et al., trained a deep neural network to identify tissue types in histological images of colorectal cancer, using a dataset compromising of over 100,000 individual patches. These patches were classified into nine distinct categories: (1) adipose tissue (ADI); (2) background (BACK); (3) debris (DEB); (4) lymphocyte (LYM); (5) mucus (MUC); (6) smooth muscle (MUS); (7) normal colon mucosa (NORM); (8) cancer-associated stroma (STR); and (9) CRC Epithelium (TUM). Their convolutional neural network model was designed to analyze the human tumor microenvironment and predict prognosis directly from histopathological images. In addition, they aggregated the abundance of each tissue type into a prognostic score, which they demonstrated to be superior in predicting survival outcomes compared to the UICC staging system, the current benchmark. They achieved an accuracy of over 94%.

Building upon Kather's methodology, Fabi Prezja et al., refined the neural network architecture initially identified by Kather and his team. They introduced a technique known as block freezing search, which played a crucial role in optimizing hyper-parameters. Their

modifications led to a remarkable increase in model accuracy, achieving 99.5% on their internal testing set and 95.6% on an external test set.

Methods

The Tensorflow colorectal_histology dataset

TensorFlow Datasets (TDFS) includes the colorectal_histology dataset under the image_classification category, which researchers use for training models to classify colorectal cancer histology images. This dataset contains a total of 5000 images and represents a collection of textures in histological images of human colorectal cancer. Each example is a 150 x 150 x 3 RGB image of one of 8 classes. The eight classes include: 0: 'tumour epithelium', 1: 'simple stroma', 2: 'complex stroma' (stroma that contains single tumour cells and/or single immune cells), 3: 'immune cell conglomerates', 4: 'debris and mucus', 5: 'mucosal glands', 6: 'adipose tissue', and 7: 'background'. All images are RGB, 0.495 μm per pixel, digitized with an Aperio ScanScope (Aperio/Leica biosystems), magnification 20x. Histological samples are fully anonymized images of formalin-fixed paraffin-embedded human colorectal adenocarcinomas (primary tumors).

Data Preparation

I used Google Colab to develop and evaluate 17 different models for classifying the 8 different textures. After loading the dataset, I converted the dataset and list into numpy arrays for easier manipulation and to enable batch processing. Then, I split the data into training and test sets. 80% of the data was used for training (4000), and the remaining 25% (1000) was used for testing. I imported 4000 examples for the training and 1000 examples for test sets and imported 4000 labels for training and 1000 labels for test sets. A 10% validation split was used. 400 examples were used for the validation set. Then, I rescaled the samples.

Methods Overview

CNNs outperform DNNs in image classification tasks. First, CNNs can learn features automatically which is highly effective for image data. Max pooling is commonly used in CNNs after convolutions layers to reduce spatial dimensions of the features maps while retaining the most important information. In addition, CNNs maintain spatial hierarchy between pixels, has translation invariance which means it can recognize a feature anywhere in the image, and has parameter sharing which can significantly reduce the number of parameters in the network. CNNs are also less prone to overfitting due to the potential for improvements with techniques such as regularization and data augmentation.

To identify the base model on which to add regularization techniques, I experimented with different neurons and hidden layers. I first experimented with different numbers of neurons in the first dense layer. This step was crucial to understand the impact of varying neuron counts on the model's performance and to find an optimal configuration. In Experiment 1a, the model architecture consisted of 128 neurons in the first dense layer. In Experiment 1b, there are 256 neurons in the first dense layer. In Experiment 1c, there are 512 neurons in the first dense layer. In Experiment 1d, there are 1024 neurons in the first dense layer.

Next, I experimented with different numbers of hidden layers to determine the most effective architecture that could capture the necessary patterns in the data. In Experiment 2a, the model architecture consisted of 2 hidden layers: the first hidden layer has 32 filters, and the second hidden layer has 64 filters. In Experiment 2b, the model architecture consisted of 3 hidden layers: the first hidden layer has 32 filters, the second hidden layer has 64 filters, and the third hidden layer has 128 filters. In Experiment 2c, the model architecture consisted of 4 hidden layers: the first hidden layer has 32 filters, the second hidden layer has 64 filters, the third hidden

layer has 128 filters, and the fourth hidden layer has 256 filters. These initial experiments helped to establish a base model architecture. Once the base model was optimized with regards to the number of neurons and hidden layers, I incorporated regularization techniques to further enhance the model's performance.

The next experiments incorporate dropout layers and explore various dropout rates. Experiment 3a incorporate dropout layers with a rate of 0.2, experiment 3b incorporate dropout layers with a rate of 0.3, and experiment 3c incorporate dropout layers with a rate of 0.4. Then, L2 regularization with various regularization rates were investigated. Experiment 4a has a regularization rate of .0001, experiment 4b has a regularization rate of .001, and experiment 4c has a regularization rate of .01. The following experiments investigate the impact of combining regularization techniques. Experiment 5a investigates the impact of dropout layers combined with L2 regularization. Experiment 5b investigates the impact of dropout layers combined with L2 regularization and batch normalization. Experiment 5c investigates the impact of dropout layers with augmentation. Finally, the last two experiments, 6a and 6b, investigate the impact of changing the optimizer to RMSprop and SGD, respectively.

Data Compilation

I used the compile() method to set up the model with Adam as the optimizer to adjust the weights. Adam combines the best properties of AdaGrad and RMSProp algorithms to handle sparse gradients on noisy problems. Adam served as the optimization algorithm to iteratively adjust weights and biases of the model based on the provided data. I used Sparse Categorial Crossentropy as the loss function to evaluate how the model is performing during training. Using categorical cross entropy as the loss function enabled effective measurement of the disparity between predicted and actual class labels. Throughout the training phase, I monitored the

accuracy metric to gauge the model's performance and track its improvement over successive epochs. For the last two experiments, I used Stochastic Gradient Descent and RMSprop as the optimizers. SGD is a variant of gradient descent where updates to the model parameters are made after each training example rather than after the entire dataset has been processed. RMSprop is an adaptive learning rate method designed to address the limitations of SGD.

Data Evaluation

I trained the models and evaluated the test and validation accuracies and losses. Then, I evaluated the performance on the test set using the evaluate method. I used Mathplotlib to create 2 plots that displayed the training and validation accuracy and loss for each epoch side by side to analyze the model's performance and inspect for overfitting. Then, I used both sklearn.metrics and the confusion matrix and to gain insight on model behavior and identify its weaknesses.

Results

Table 1 shows all the accuracies and losses for the training, validation, and test sets as well as the performance time for all 17 neural network models investigated in this study. For the first set of experiments, Model 1a to Model 1d, the best performing neural network featured 128 neurons in the first dense layer, with an accuracy of 74.9%. The addition of more neurons in the first dense layer failed to improve model performance. Therefore, 128 neurons were used for next set of experiments, Model 2a to Model 2c. The best performing neural network from this set has 3 hidden layers. As a result, the base model has 3 hidden layers (filters = 32, 64, 128) and 128 units in the first dense layer.

The next set of experiments, Model 3a to Model 3c, added dropout layers and tested dropout rates of 0.2, 0.3, and 0.4. The best performing model featured a dropout rate of 0.2, with an accuracy of 77.8 %. Model 4a to Model 4c added L2 regularization and tested regularization

rates of .0001, .001, and .01. The best performing model featured a regularization rate of .0001, with an accuracy of 76.2%.

The combination of dropout layers (0.2) and L2 regularization (.001) provided an accuracy rate of 73%, which did not outperform the best model. The addition of batch normalization decreased the accuracy significantly, leading to a rate of 45.4 %. The combination of dropout layers (0.2) with augmentation resulted in an accuracy rate of 76.2%, slightly less than the best performing model. Model 17 which used RMSprop resulted in an accuracy of 64%. Model 18 which used SGD resulted in an accuracy of 43.1%. The Adam optimizer still surpassed both optimizers.

Conclusions

The model with the highest accuracy was the CNN with 3 hidden layers, 128 hidden nodes, and no regularization, achieving 78.2% accuracy. However, visualizations showed that the training and validation set values did not match as closely compared to model from Experiment 8. Although the model from Experiment 8 achieved a slightly lower accuracy, the graphs indicate better generalization and less overfitting. Therefore, the results demonstrate that a convolutional neural network with 3 hidden layers and dropout layers with 0.2 dropout rates outperformed the rest of the models.

The addition of other regularization techniques failed to improve model performance, possibly due to the small dataset size. In such cases, regularization can sometimes prevent models from capturing the underlying patterns effectively. In other words, a model might need to use all the available information to learn patterns, and regularization can inhibit this process.

My goal is to develop a tool that can classify histology images using the colorectal_histology dataset, which consists of 5000 images. The best model I developed attained

an accuracy of 77.8%. My recommendation for management is to use convolutional neural networks with 3 hidden layers and incorporate regularization techniques, such as dropout layers with rates of 0.2. However, compared to the literature, it is evident that a larger dataset can significantly impact model performance. Model development is always data dependent, and a larger dataset can provide models with greater insights into underlying patterns. I recommend researchers to repeat these experiments on a larger dataset and to aim for an accuracy of over 95%. This high accuracy is crucial in medical diagnostics, given the significant impact on patient outcomes.

References

"Colorectal Histology Dataset." TensorFlow. Accessed June 10, 2024. https://www.tensorflow.org/datasets/catalog/colorectal_histology.

Kather, Jakob Nikolas, et al. "Predicting Survival from Colorectal Cancer Histology Slides Using Deep Learning: A Retrospective Multicenter Study." *PLoS Medicine* 16, no. 1 (January 24, 2019): e1002730. https://doi.org/10.1371/journal.pmed.1002730.

Prezja, Fabi, et al. "Improved Accuracy in Colorectal Cancer Tissue Decomposition Through Refinement of Established Deep Learning Solutions." *Scientific Reports* 13, no. 1 (September 23, 2023): 15879. https://doi.org/10.1038/s41598-023-42357-x.

Appendix

TABLE 1. SUMMARY: PERFORMANCES FOR ALL 17 ARCHITECTURES

	Train		Test		Validation		
Experiment	Accuracy	Loss	Accuracy	Loss	Accuracy	Loss	Process Time
1	0.9464	0.1923	0.749	0.7334	0.6625	0.9059	2:43
2	0.9122	0.2969	0.691	0.7356	0.7175	0.749	2:00
3	0.9789	0.0844	0.739	0.6635	0.76	0.7263	3:44
4	0.8217	0.4557	0.68	0.8234	0.7275	0.8076	3:51
5	0.9164	0.2505	0.739	0.6724	0.72	0.8571	1:44
6	0.9211	0.2477	0.782	0.6277	0.765	0.9581	3:26
7	0.7794	0.559	0.769	0.598	0.7675	0.5742	2:51
8	0.8575	0.3697	0.778	0.6178	0.8025	0.5736	4:25
9	0.7508	0.6424	0.729	0.7081	0.58	1.0196	2:37
10	0.7381	0.6317	0.668	0.6558	0.69	0.7187	3:47
11	0.8172	0.5202	0.762	0.683	0.76	0.7356	1:55
12	0.8144	0.6574	0.747	0.8044	0.775	0.8531	2:00
13	0.7558	0.9183	0.745	0.9791	0.7825	0.9394	3:02
14	0.7633	0.6428	0.73	0.7814	0.6925	0.7891	2:22
15	0.8175	0.5253	0.454	2.2388	0.375	2.1345	2:21
16	0.8689	0.3373	0.762	0.6042	0.7525	0.8821	2:10
17	0.7267	0.7103	0.64	0.9809	0.635	0.9362	1:56
18	0.5436	1.059	0.431	1.1651	0.4125	1.6594	1:55

NEURAL NETWORK ARCHITECTURES:

Experiment 1: Model 1a: CNN with 1 convolution/max pooling layers (no regularization). filters=32, **first dense layer units=128**

Experiment 2: Model 1b: CNN with 1 convolution/max pooling layers (no regularization). filters=32, **first dense layer units=256**

Experiment 3: Model 1c: CNN with 1 convolution/max pooling layers (no regularization). filters=32, **first dense layer units=512**

Experiment 4: Model 1d: CNN with 1 convolution/max pooling layers (no regularization). filters=32, **first dense layer units=1024**

Experiment 5: Model 2a: Model 2a: CNN with 2 convolution/max pooling layers (no regularization). **filters=32, 64.** first dense layer units=128

Experiment 6: Model 2b: CNN with 3 convolution/max pooling layers (no regularization). **filters=32, 64, 128.** first dense layer units=128

Experiment 7: Model 2c: CNN with 4 convolution/max pooling layers (no regularization). **filters=32, 64, 128, 256.** first dense layer units=128

Experiment 8: Model 3a: CNN with 3 convolution/max pooling layers **(dropout=0.2)**. filters=32, 64, 128. first dense layer units=128

Experiment 9: Model 3b: CNN with 3 convolution/max pooling layers **(dropout=0.3).** filters=32, 64, 128. first dense layer units=128

Experiment 10: Model 3c: CNN with 3 convolution/max pooling layers **(dropout=0.4).** filters=32, 64, 128. first dense layer units=128

Experiment 11: Model 4a: CNN with 3 convolution/max pooling layers **(L2 regularization=0.0001).** filters=32, 64, 128. first dense layer units=128

Experiment 12: Model 4b: CNN with 3 convolution/max pooling layers **(L2 regularization=0.001).** filters=32, 64, 128. first dense layer units=128

Experiment 13: Model 4c: CNN with 3 convolution/max pooling layers **(L2 regularization=0.01).** filters=32, 64, 128. first dense layer units=128

Experiment 14: Model 5a: CNN with 3 convolution/max pooling layers **(drop out 0.2 and L2 regularization 0.0001).** filters=32, 64, 128. first dense layer units=128

Experiment 15: Model 5b: CNN with 3 convolution/max pooling layers **(drop out 0.2 and L2 regularization .0001 and batch normalization).** filters=32, 64, 128. first dense layer units=128

Experiment 16: Model 5c: CNN with 3 convolution/max pooling layers **(drop out 0.2 and augmentation).** filters=32, 64, 128. first dense layer units=128

Experiment 17: Model 6a: CNN with 3 convolution/max pooling layers **(no regularization, optimizer='RMSprop').** filters=32, 64, 128. first dense layer units=128

Experiment 18: Model 6b: CNN with 3 convolution/max pooling layers **(no regularization, optimizer='SGD').** filters=32, 64, 128. first dense layer units=128

Figure 1: Architecture of the CNN from Experiment 8

Layer (type)	Output Shape	Param #
conv2d_13 (Conv2D)	(None, 148, 148, 32)	896
max_pooling2d_13 (MaxPooling2D)	(None, 74, 74, 32)	0
dropout (Dropout)	(None, 74, 74, 32)	0
conv2d_14 (Conv2D)	(None, 72, 72, 64)	18496
max_pooling2d_14 (MaxPooling2D)	(None, 36, 36, 64)	0
dropout_1 (Dropout)	(None, 36, 36, 64)	0
conv2d_15 (Conv2D)	(None, 34, 34, 128)	73856
max_pooling2d_15 (MaxPooling2D)	(None, 17, 17, 128)	0
dropout_2 (Dropout)	(None, 17, 17, 128)	0
flatten_7 (Flatten)	(None, 36992)	0
dense_14 (Dense)	(None, 128)	4735104
dropout_3 (Dropout)	(None, 128)	0
dense_15 (Dense)	(None, 8)	1032

Figure 2: Training set and validation set accuracy and loss from Experiment 8

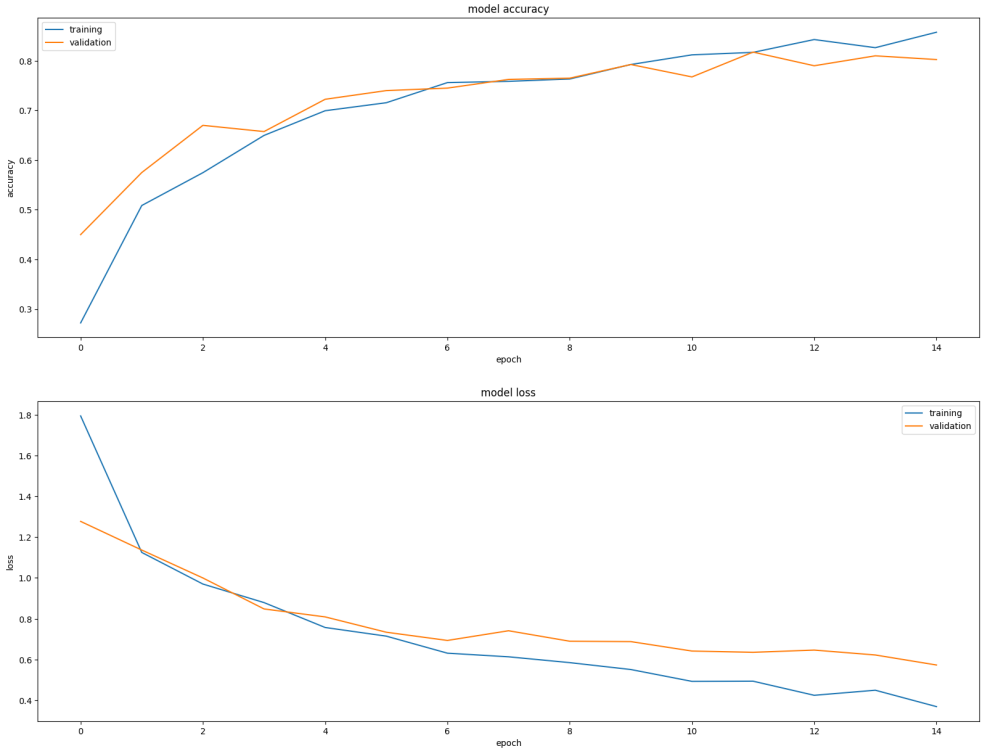

Figure 3: Confusion Matrix from Experiment 8

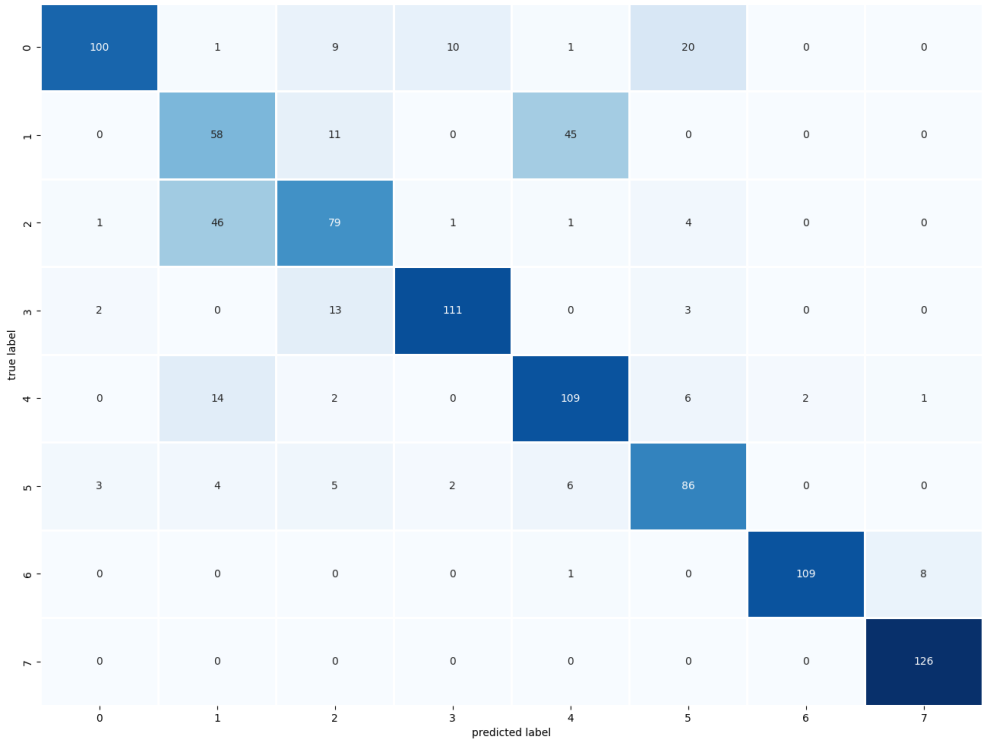

Figure 4: TSNE for Experiment 8