Mouse Brain Challenge: Strategy and Results

Junting Ren

February 26, 2023

Readme for the code

- All the workhorse codes are in the python script files (.py)
- The Jupiter notebook import these scripts for training and evaluation.
- Open the main.ipynb, it contains two big parts:
  - Training the model from scratch. For this to work, you will need to input the path to the data folder that is attached in the mail (the folder containing 3 sub-folders of the names of the slices).
  - Evaluating the model using saved weights. The input is a (N, 512, 512, 3) numpy array as instructed.

Overview

- Problem setup and dataset description
- Challenges
- Model architecture and pretraining
- Data augmentation and fine-tuning strategy
- Results and interpretation
- Conclusion

Problem setup and dataset description

- The problem is to classify the mouse brain images into three different slice views: Horizontal, Coronal and Sagittal.
- I am using the default dataset attached in the mail from the BrainMap atlas.
- This dataset contains: 340 Horizontal, 322 Coronal and 374 Sagittal images.
- The images have various resolution and stains (color).

Challenges

- The 3 different categories' sample size is balanced. So no need to reweight the instances.
- On the first glance, it seems easy to classify the images purely based on shape. But there are edge cases where the shape is indistinguishable between classes.
- The variance of the prediction may be high, since if the validation set does not contain different stains and resolutions within each category, then the training could be stopped early due to the validation set's loss ceasing to decrease.

Model Architecture and pretraining

- Since the task is relatively simple and there are only three category, it is sensible to select a relative shallow network to save training time and computational power.
- Therefore, the strategy is to start with ResNet containing 18 layers and see if the performance is good. If not, we increase the depth.
- ResNet residual connections [3] enjoys faster convergence if we need more depth, and the adaptation to various input size is also a plus.
- MedNet [2] provide pretrained ResNet18 on medical X-ray images, but our task is more focus on capturing the shape similar to ImageNet task [1]. Therefore, I chose the ResNet18 pretrained on ImageNet for my starting point.
- The final fully connected layer is replaced with a layer outputting 3 Logit scores.

Data train, validation and test splitting

- The data is split into training, validation and testing set. The training set contains 70% of the full dataset, and both validation and testing set contains 15% of the full dataset.
- To make the the categories balanced between the categories, the train, validation and testing set is divided within each category and combined.
- The dataset is shuffled before splitting, so it is assumed that the various stain and resolution images will be equally divided between the different sub-sets.
- I set the random seed to be 2 for reproducibility.
- The training dataset contains: 238 Horizontal, 226 Coronal, 262 Sagittal. The validation dataset contains: 51 Horizontal, 48 Coronal, 56 Sagittal. The testing dataset contains: 51 Horizontal, 48 Coronal, 56 Sagittal.

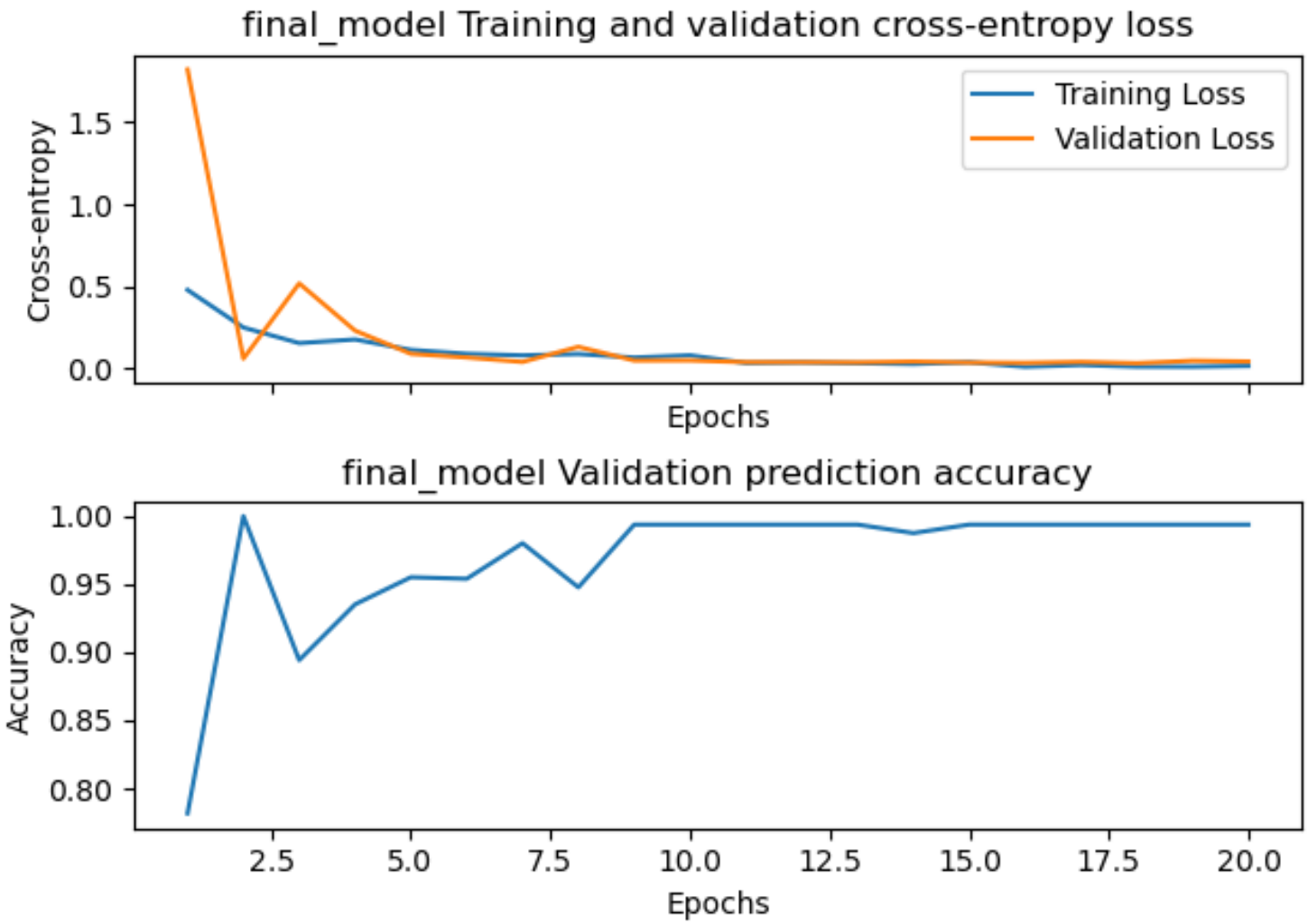
Data pre-processing and augmentation

- The mean and standard deviation of the training images for the 3 color channels is calculated and saved.
- Both the training, validation and testing images are first re-scaled to [0,1], resized to have the same size of [512, 512] and each channel is normalized using the training images' mean and standard deviation.
- Since we want the model to capture not only the shape but also the textures of the image, random crop of at most 50% and random rotation of at most 180 degrees help with this goal.
- Therefore, I added random crop and random rotation each with probability of 0.5 applying to each training image. Validation and testing images are not augmented.

Fine-tuning strategy

- All weights in ResNet18 are tuned using Adam stochastic gradient descent algorithm that maintain both momentum and adptive learning rate for different parameters.
- My code can easily be modified to test out different parameter setup through adding parameters in the config dict in the Jupiter notebook, and each setup result will be automatically saved to the experimentation folder.
- I used the cross-entropy as the loss. The model is trained on the training dataset and validated on the validation set for a maximum of 20 epochs. If the validation set's loss decreases comparing to the current smallest validation loss within most recent 5 epochs, the training continues until reaches the maximum epoch of 20.
- The testing images are the held-out set where the model has never seen. It is used to evaluate the model's final performance.

Training and validation loss plot

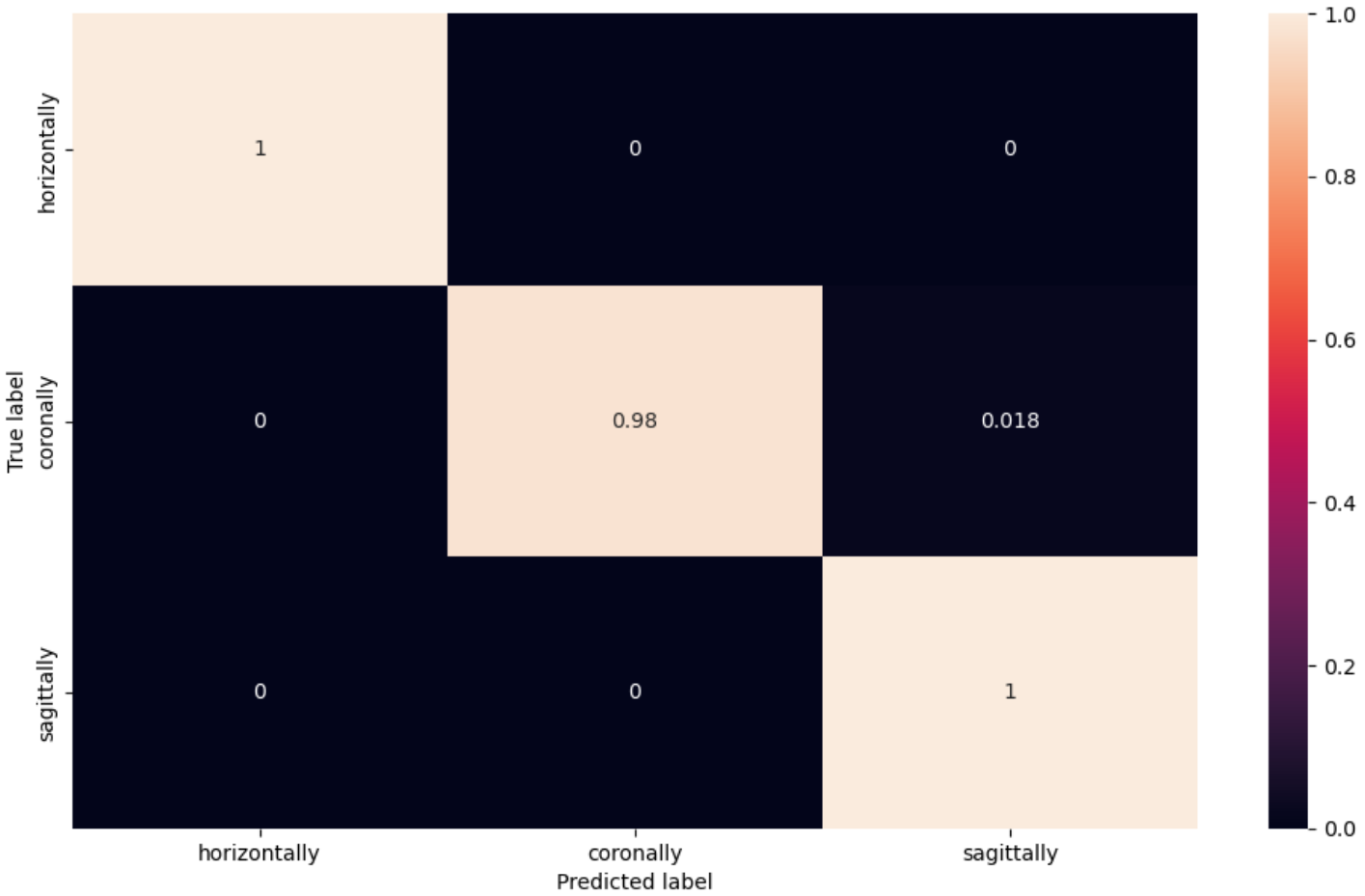


Fine-tuning parameter

- I did not search for the optimal parameter since on the first try, the model achieved 99% testing accuracy.
- The model with the lowest validation error is saved and used for the final prediction on the test dataset.
- The hyper-parameters I used for the final model are:
  - Learning rate: 0.001
  - l2 penalize strength 0.0005
  - 0.9 exponential decay rate for the learning rate

Results

The model achieved overall of 99% accuracy, and below is the confusion matrix for the test dataset prediction:



Visualizing the mistake using GradCAM

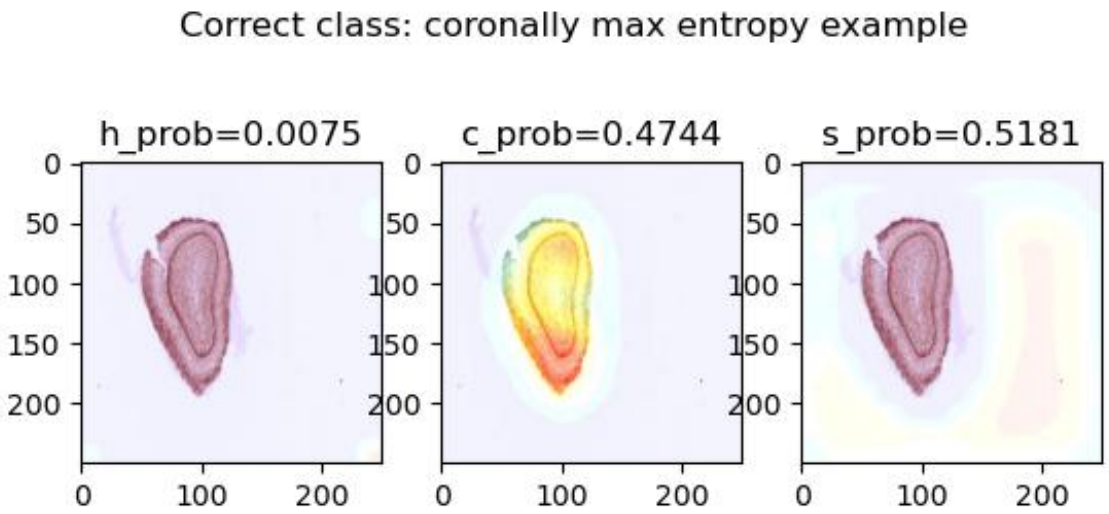


Figure: The correct class is coronally. h_prob: the probability of predicting horizontal. c_prob and s_prob are the probabilities of predicting coronally and sagittally.

Visualizing the mistake using GradCAM

- I used the activation of the final 14 by 14 convolution layer to visualize what the model is focusing on making the prediction for the specific class.
- The model made one prediction mistake for the true category of coronally in the test dataset.
- As seen in Figure 1, this is an edge case of coronally slice where only one side is visible, and there are some artifacts on the right hand side where the model is focusing on for predicting the wrong class sagittally.
- Solution: by increasing the sample size so that this edge cases is included in the training dataset will correct this mistake.

Interpreting the model

- I selected 1 random cases for all three categories to visualize what feature the model is focusing on to make the predictions as shown in the next few slides from Figure 2, 3 and 4.
- We can see for both horizontal and coronally slice, ResNet mainly captures the shape characteristics of the slices as shown in Figure 2 and 3.
- What is interesting is that for sagittally slice, the model is not only capturing the shape but also the distinct texture on the right upper part of the image as shown in Figure 4.

Random selected image for Horizontal slice

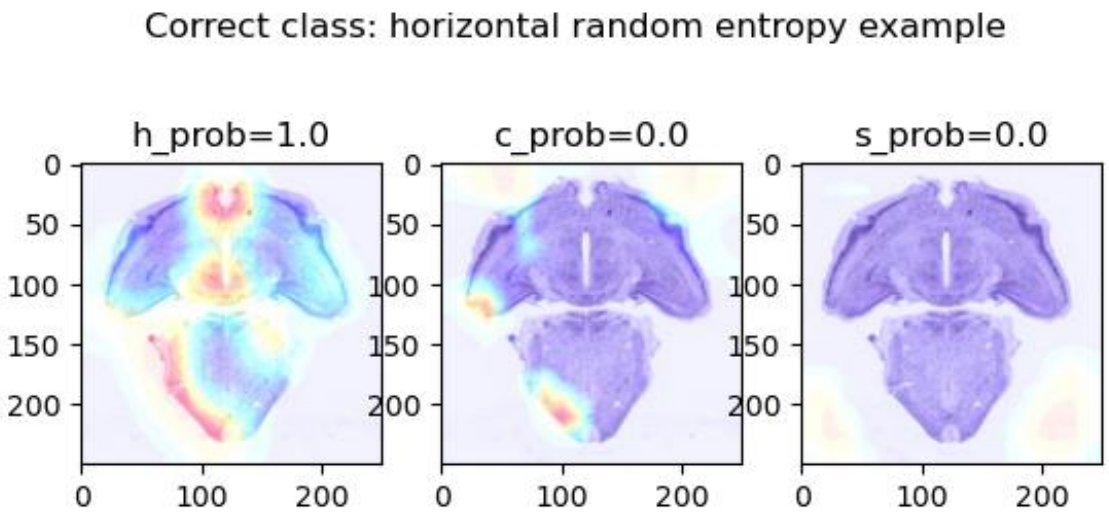


Figure: The correct class is horizontal. h_prob: the probability of predicting horizontal. c_prob and s_prob are the probabilities of predicting coronally and sagittally.

Random selected image for coronally slice

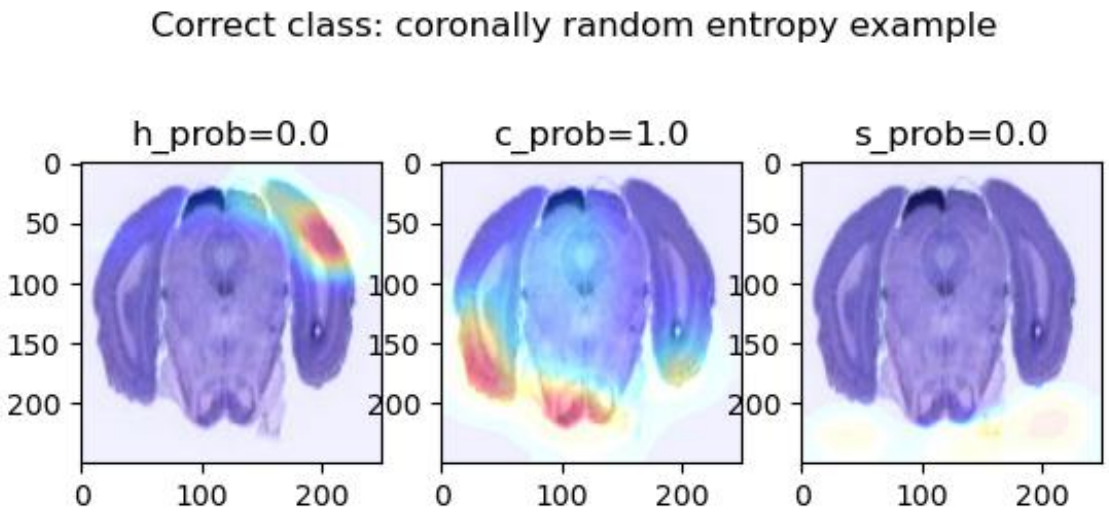


Figure: The correct class is coronally. h_prob: the probability of predicting horizontal. c_prob and s_prob are the probabilities of predicting coronally and sagittally.

Random selected image for sagittally slice

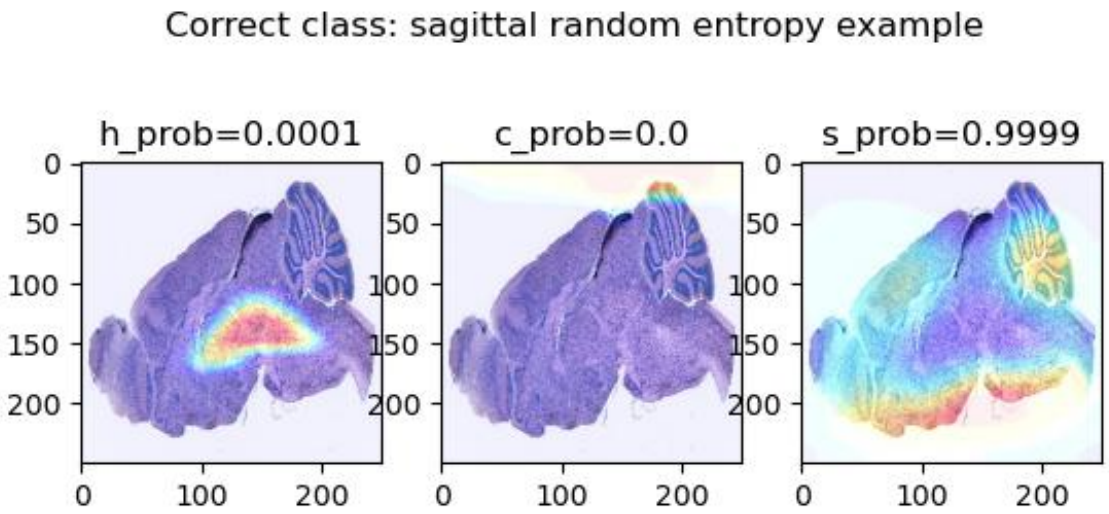


Figure: The correct class is sagittally. h_prob: the probability of predicting horizontal. c_prob and s_prob are the probabilities of predicting coronally and sagittally.

Conclusion

- The shallow ResNet performances very well on this relatively simple task.
- To avoid over-fitting, I only continue on training the model if the validation loss is decreasing. To avoid bias in the evaluating the performance of the model, a test dataset is created for which the model has never seen in training.
- Using random crop and rotation image augmentation during training, the ResNet not only capture the shape but also the textures of the different slices.

End

Thank you!

[1] Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee.

[2] Guo, Y., Camino, A., Wang, J., Huang, D., Hwang, T. S., and Jia, Y. (2018). Mednet, a neural network for automated detection of avascular area in oct angiography. *Biomedical optics express*, 9(11):5147–5158.

[3] He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.