Cardiovascular risk computed via Deep Learning (DL) on thoracic CT scans (Med3DResNet)

Timothy Burt, Luben Popov, Yuan Zi

COSC 7373 Adv. Computer Vision F19 Team 1

3D CNN: Ontology and 'ResNets'

We employ an ResNet-like architecture similar to that used in [1].

The original paper on ResNets from 2016 [2] successfully argues that "residual blocks" have a better accuracy and speed than standard CNN blocks which do not have a residual mapping function connecting adjacent blocks.

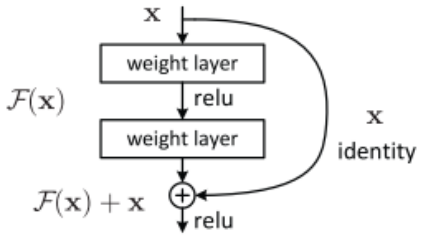

Figure 1: Residual learning: a building block [2]

While the source code was based on a ResNet-TensorFlow implementation on GitHub, it has been extensively modified to work with 3D medical scans. Specifically our input size is 256X256X256 pixels, with a 16-bit integer intensity in Hounsfield Units.

Our Med3DResNet leaves the X and Y image dimensions alone for each slice, but averaging is performed in the z-direction to compress unneeded data. The code currently allows for 4, 8, and 16 slice axial-averages and this information is then fed into the number of channels for each input image (64, 32, and 16 channels, respectively).

Network architecture graphs for 3 of our selected models are shown below, generated using TensorBoard.

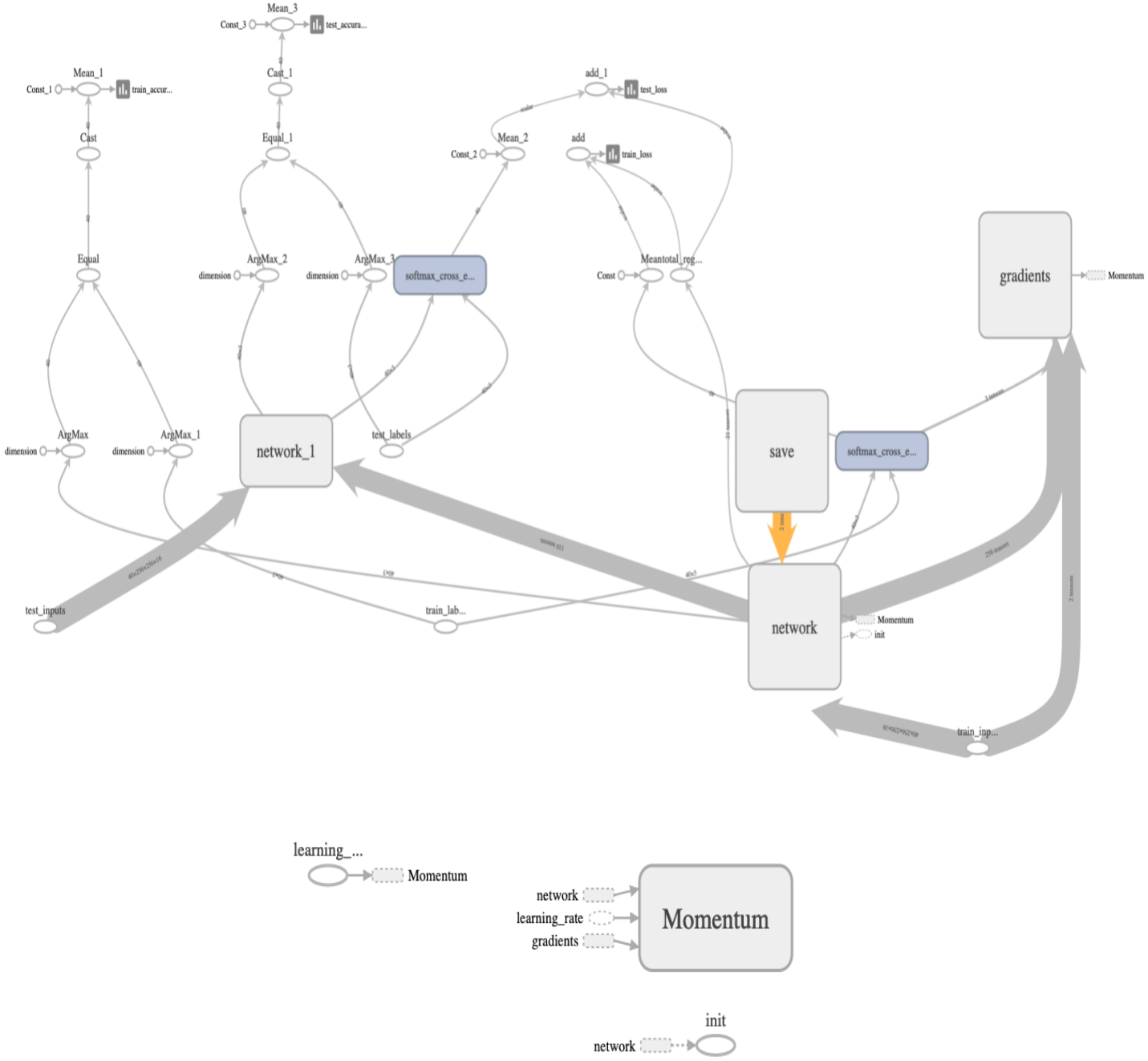

Figure 2: Op-level TensorBoard graph of ResNet 18-layer network. Residual blocks [2, 2, 2, 2]. 16 z-slice averaged (16 CH).

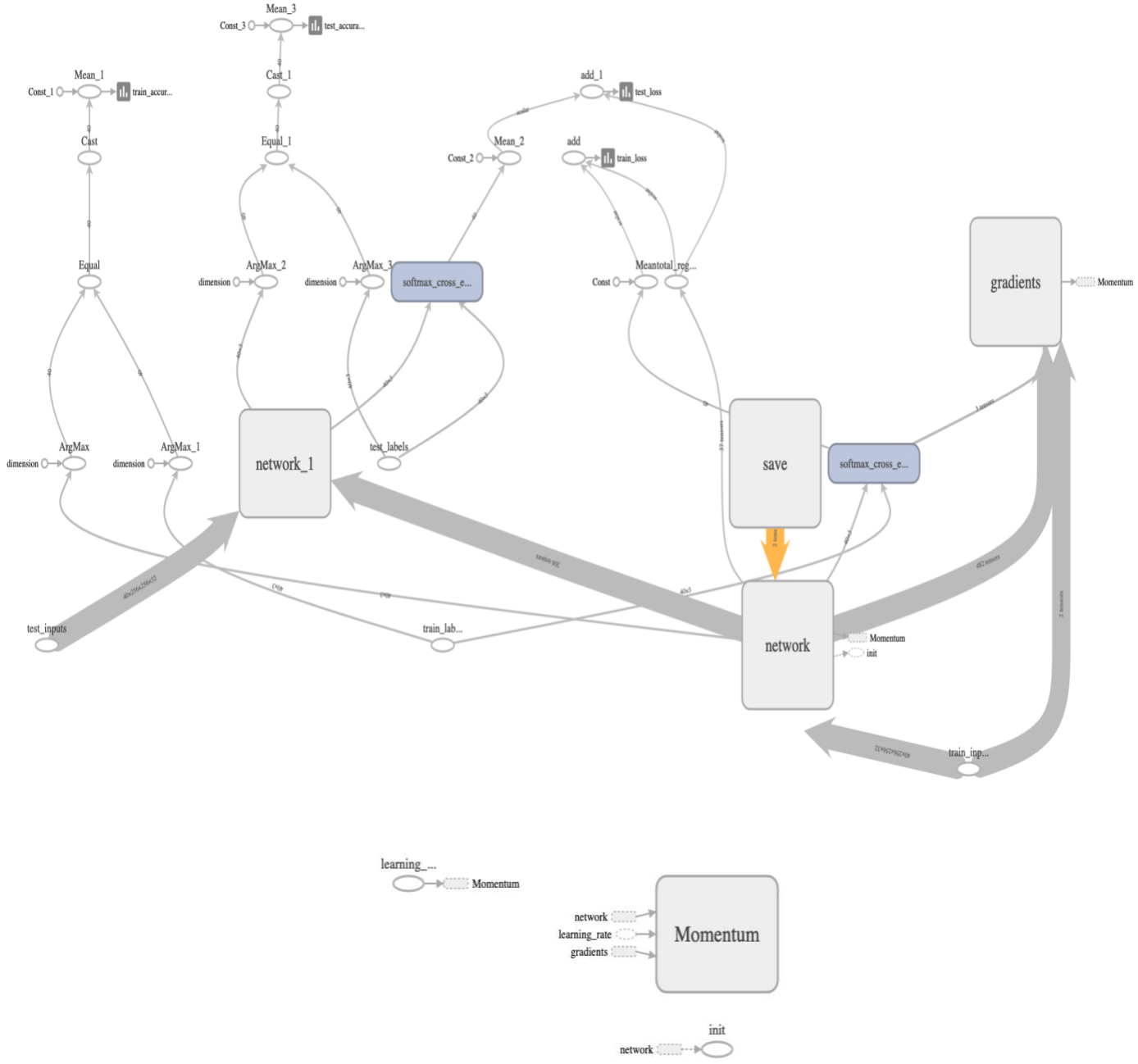

Figure 3: Op-level TensorBoard graph of ResNet 34-layer network. Residual blocks [3, 4, 6, 3]. 8 z-slice averaged (32 CH).

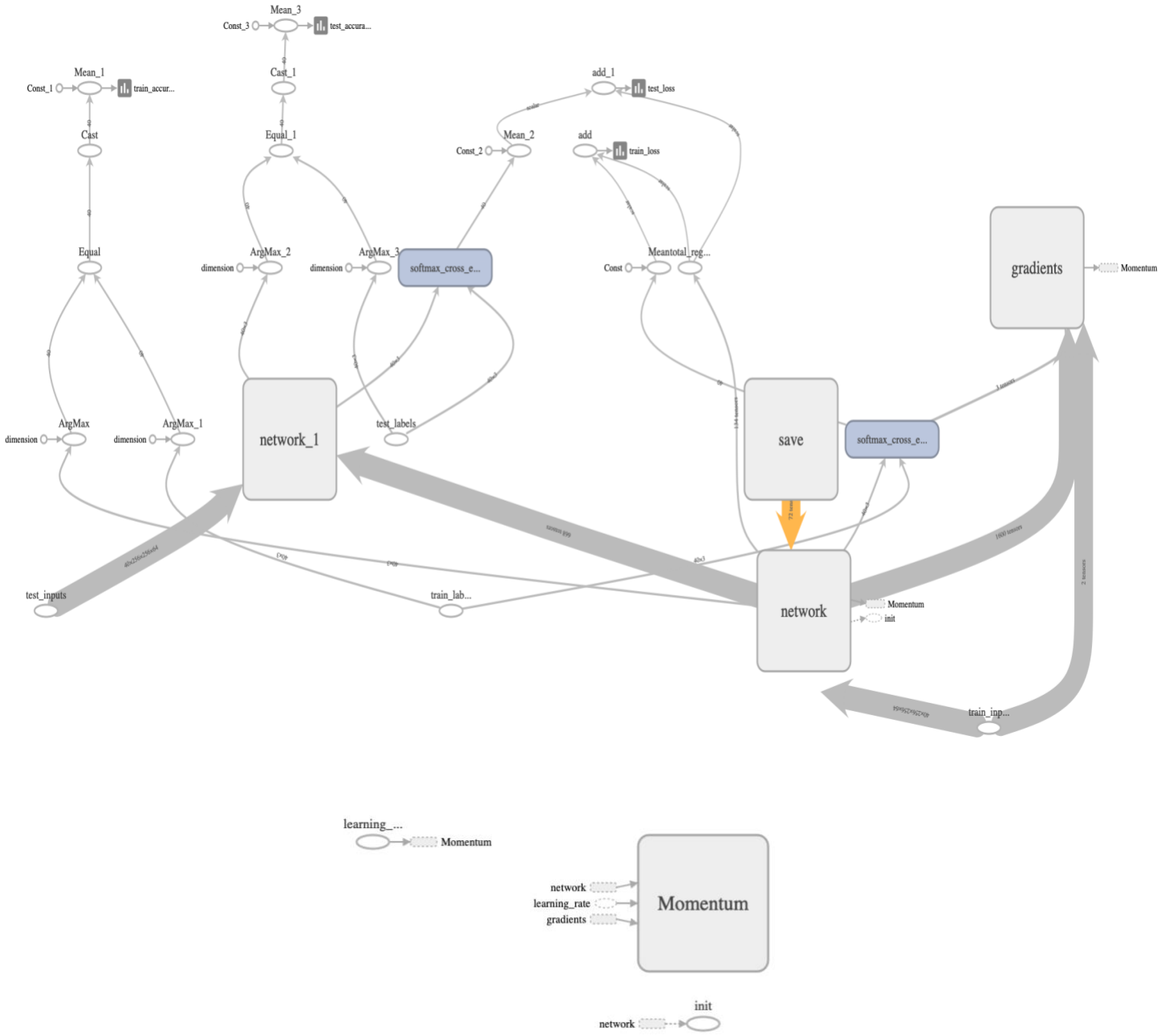

Figure 4: Op-level TensorBoard graph of ResNet 101-layer network. Residual blocks [3, 4, 23, 3]. 4 z-slice averaged (64 CH).

LIDC-IDRI Dataset [3]

- 1,017 thoracic CT scans
  - Subset of 157 patients have diagnosis data, all annotated with our tool
  - 4 classes:
    1) unknown
    2) benign or non-malignant disease
    3) malignant, primary lung cancer
    4) malignant metastatic (spread beyond the lungs)

The unknown class could not be used because a confirmation is not provided, even though a diagnosis is. So the stage of the disease is unknown.

Therefore, 3 latent classes are used from the diagnostic truth containing 130 annotated CT scans. Below is a table with our train/test validation method.

Train/validation method

To eliminate bias and obtain the best test accuracy with our limited dataset, 70/30 and 80/20 train/test split .csv files are provided, and either may be called using a flag in our tool. Each class is split evenly be hand to ensure an equal split for each class. Random shuffling is performed within each group as well during every epoch.

	Train	Test	Train	Test	Total Annotated Data
	0.7	0.3	0.8	0.2	
Benign	25	11	28	8	36
Malignant, primary lung cancer	30	13	34	9	43
Malignant metastatic (spreading)	35	16	40	11	51
Total Annotated Data	90	40	102	28	130

Table 1: Train/test distributions among 3 latent classes for our annotated "diagnostic truth" CT images

3D CNN tutorial

To start the training/testing process, it is assumed you have already annotated the relevant patients and batch exported the affine/projection images.

CNN_main.py is the entry point for training/validation of our Med3DResNet. There are a number of flags which must be provided at runtime which contain the model specifications, given in the table below.

Flag	Type	Definition
phase	str	Train or test. Train will do initial training, and test will give a final accuracy f-score loading the model.
epoch	int	Number of epochs to run. 25 works well.
batch size	int	Determines k-fold CV k value. Must be less than total train/test data samples. (e.g. with 100 data samples, if batch_size is 50 and total data size is 100, each epoch will split the data randomly into 2 folds for cross validation.
res_n	int	Number of residual layers for model. '18, 34, 50, 101, 152' may all be called, each doubling in memory requirements as they increase.
lr	float	Learning rate. This is an adaptive hyperparameter, starting at 0.1 by default and slows to 0.001 by the end of training.
use_lung_mask	action_bool	If flag provided, the model will read masked images of the specified type which remove the lungs (only works with affine correctly for now)
data_type	str	Either `affine` or `projection` generated images can be used to train the CNN
n_axial_channels	int	Compression of z-direction axial slice thickness [256 -> 16,8,4]. This also determines the input image channels [1 -> 16,32,64].
train_test_ratio	str	Training/testing data split `70_30` or `80_20`.

Table 2: CNN_main.py flags relevant to model implementation with data type and description

Checkpoint and log folder paths can be specified, and this is where model snapshots are stored every epoch along with a log. The subfolder names are unique to each model you can run (e.g. *ResNet18_ACVaffine_lungmaskFalse_ch16_traintest_70_30_40_0.1*).

In case the training process is stopped, this allows the program to pick up from where it left off. This folder contains the weights within each layer of the NN, and simply calling `tensorboard` from within the log folder allows automatic network graphs and training/testing accuracy/loss data to be exported and graphically viewed.

Validation is performed at the end of the training, so the `test` phase can be used to redo and output the f-accuracy score for the model at a later time.

Example CNN train/test output

(ACVProject) timothyburt@Timothys-iMac:~/PycharmProjects/ACVProject$ python CNN_main.py –phase train –dataset ACV –res_n 18 –train_test_ratio 70_30 –batch_size 40 –lr 0.1 –data_type projection –n_axial_channels=4 –epoch 25

Using TensorFlow backend.

Loading diagnostic truth class labels…

70_30 train/test ratio being used…

Loading/sorting image data into test/train split from .csv files…

Loading projection images…

Averaging z-block 1 of 64… # doesn't take over 1 min.

…

Averaging z-block 64 of 64…

Normalizing z-blocks…

Variables: name (type shape) [size] # this outputs the ResNet network model details, explicitly.

network/conv/conv2d/kernel:0 (float32_ref 3x3x64x32) [18432, bytes: 73728]

…

network/logit/dense/bias:0 (float32_ref 3) [3, bytes: 12]

Total size of variables: 2814883

Total bytes of variables: 11259532

4-slice axial-averaged ResNet CNN…

[*] Reading checkpoints…

[*] Failed to find a checkpoint

[!] Load failed…

Epoch: [0] [0/ 2] time: 32.4900, train_accuracy: 0.25, test_accuracy: 0.38, learning_rate : 0.1000

Epoch: [0] [1/ 2] time: 61.2496, train_accuracy: 0.40, test_accuracy: 0.38, learning_rate : 0.1000

Epoch: [1] [0/ 2] time: 90.2116, train_accuracy: 0.55, test_accuracy: 0.38, learning_rate : 0.1000

Epoch: [1] [1/ 2] time: 119.1739, train_accuracy: 0.50, test_accuracy: 0.30, learning_rate : 0.1000

Epoch: [2] [0/ 2] time: 147.9583, train_accuracy: 0.57, test_accuracy: 0.35, learning_rate : 0.1000

Epoch: [2] [1/ 2] time: 176.7999, train_accuracy: 0.57, test_accuracy: 0.38, learning_rate : 0.1000

Epoch: [3] [0/ 2] time: 205.9900, train_accuracy: 0.62, test_accuracy: 0.38, learning_rate : 0.1000

Epoch: [3] [1/ 2] time: 234.7345, train_accuracy: 0.60, test_accuracy: 0.32, learning_rate : 0.1000

Epoch: [4] [0/ 2] time: 263.8747, train_accuracy: 0.70, test_accuracy: 0.35, learning_rate : 0.1000

Epoch: [4] [1/ 2] time: 292.5193, train_accuracy: 0.70, test_accuracy: 0.38, learning_rate : 0.1000

Epoch: [5] [0/ 2] time: 321.7517, train_accuracy: 0.73, test_accuracy: 0.38, learning_rate : 0.1000

Epoch: [5] [1/ 2] time: 350.3955, train_accuracy: 0.73, test_accuracy: 0.38, learning_rate : 0.1000

Epoch: [6] [0/ 2] time: 379.1660, train_accuracy: 0.77, test_accuracy: 0.38, learning_rate : 0.1000

Epoch: [6] [1/ 2] time: 407.9822, train_accuracy: 0.70, test_accuracy: 0.40, learning_rate : 0.1000

Epoch: [7] [0/ 2] time: 437.2588, train_accuracy: 0.80, test_accuracy: 0.38, learning_rate : 0.1000

Epoch: [7] [1/ 2] time: 466.0439, train_accuracy: 0.77, test_accuracy: 0.45, learning_rate : 0.1000

Epoch: [8] [0/ 2] time: 494.8529, train_accuracy: 0.82, test_accuracy: 0.45, learning_rate : 0.1000

Epoch: [8] [1/ 2] time: 523.8370, train_accuracy: 0.80, test_accuracy: 0.43, learning_rate : 0.1000

…

Epoch: [24] [1/ 2] time: 1466.0858, train_accuracy: 0.75, test_accuracy: 0.38, learning_rate : 0.0010

[*] Training finished!

[*] Reading checkpoints…

[*] Success to read ResNet.model-51

[*] Load SUCCESS

test_accuracy: 0.375

[*] Test finished!

Med3DResNet Results

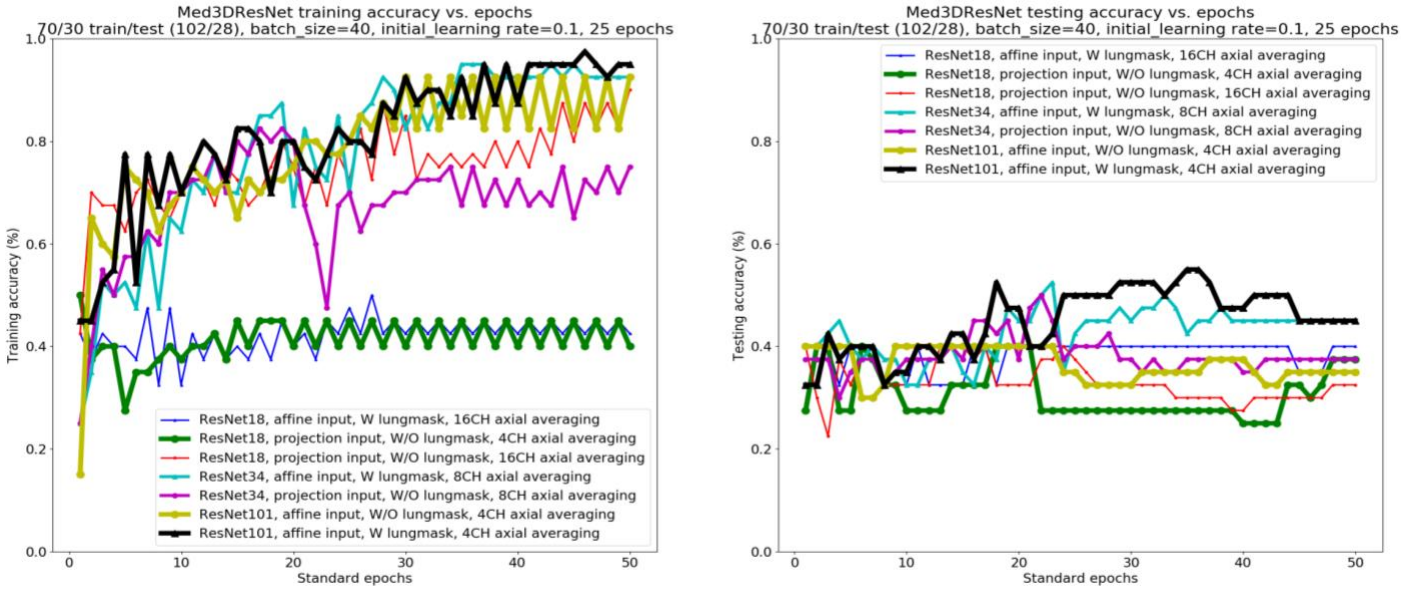

Figure 5: Med3DResNet training accuracy vs. epochs (left), testing accuracy vs. epochs (right) for several models/inputs.

Model	Test accuracy (2-fold CV)
ResNet18, affine input, W lungmask, 16CH axial averaging	0.325
ResNet18, projection input, W/O lungmask, 4CH axial averaging	0.375
ResNet18, projection input, W/O lungmask, 16CH axial averaging	0.450
ResNet34, affine input, W lungmask, 8CH axial averaging	0.350
ResNet34, projection input, W/O lungmask, 8CH axial averaging	0.450
ResNet101, affine input, W/O lungmask, 4CH axial averaging	0.400
ResNet101, affine input, W lungmask, 4CH axial averaging	0.375

Table 3: Comparison of testing accuracy measured from 2-fold CV for 7 models.

Above are the training and testing accuracy vs. standard epochs (epochs*k, where k=2 in k-fold cross validation (CV)). Due to data and time constraints, we were not able to test other variants of our model.

However, there are several features worth noting in this plot which give insight into which network architecture and input data normalization provide the best testing accuracy.

The ResNet 101-layer network trained on 4 z-slice averaged blocks (64 CH) did give a test accuracy higher than chance for some of the epochs, and was the only model to do so within the limited amount of training time used.

It seems that the projection-shaped inputs, which fill the entire 256x256 slice while preserving scale, in every model tested gave almost 10% better test accuracy than with affine-shaped inputs, which left much wasted space in the data.

Applying a lung mask may or may not help for affine (more data needed), but below are two images showing how the mask works to remove features not within the heart.

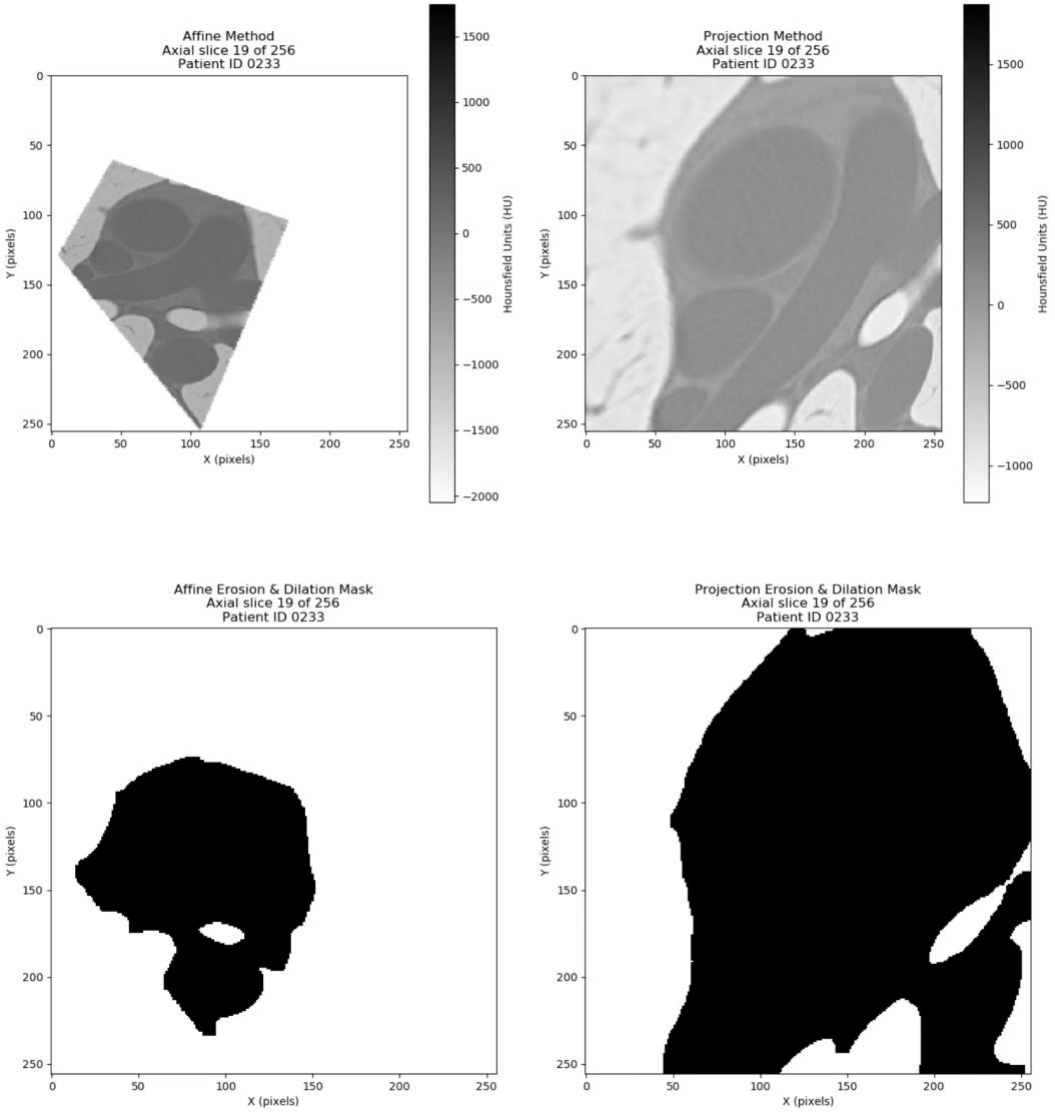

Figure 6: Axial slice showing affine method (top left) vs. projection method (top right), and their corresponding masks generated via an automated k-means clustering and erosion & dilation mask method.

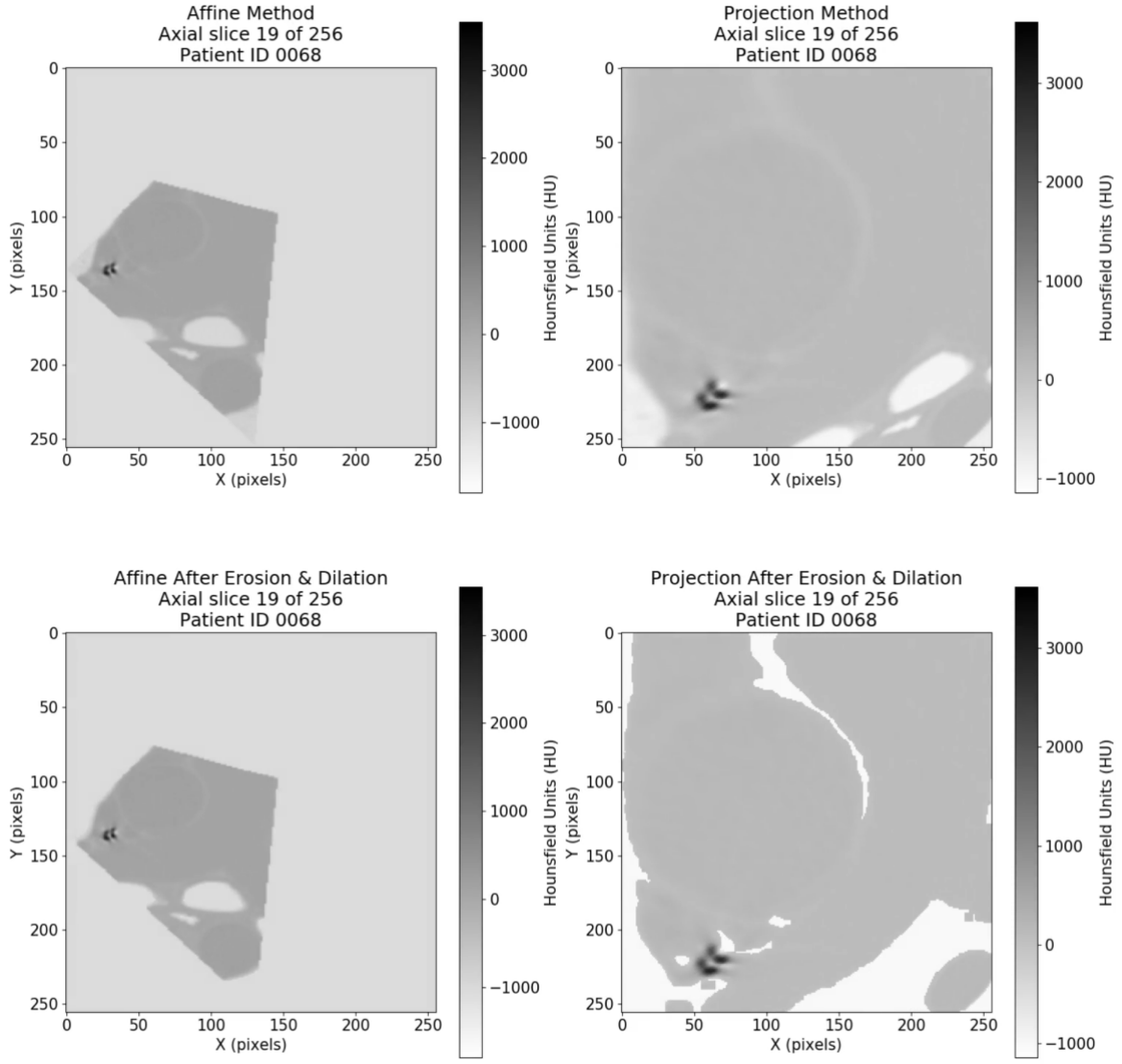

Figure 7: Axial slice showing affine method (top left) vs. projection method (top right), and their corresponding masks applied to them to remove structures not part of the heart.

While our automated method worked fine for affine inputs, it needed more fine-tuning to work for the projection inputs, as can be seen above. This is why we did not test CNN models with projection input and with lung masks applied.

It is interesting that our z-slice averaging into the input channels helps models with less layers (like ResNet18) to perform better, but with large averaging (16 slice). Thus ResNets that have higher training accuracy and less hidden layers find larger features within the chest, and these would not find smaller features associated with early-stage atherosclerosis.

Because of this, it appears that **deep and wide** CNNs are needed to detect features in CT scans that have a medical interpretation which is necessary.

Coronal arteries can measure as small as 3mm, about the thickness of a single axial slice with no averaging over neighboring ones. Due to this, it would be simple to add a `None` flag to the n_axial_channels flag and provide all 256 z-slices into a ResNet 101 architecture with the z-axis as the channels, allowing us to see how z-direction averaging & binning truly affects accuracy.

This would add considerably to memory requirements however, as training a ResNet101 on a 2019 iMac using TensorFlow on the CPU takes 64GB+ of memory if available.

Caveats

First, we performed a few models using 80/20 train/test split as outlined in the above section, but test accuracy results were about half of what we obtained with the 70/30 split. This confirms that our lack of data is the largest factor in the low accuracy of our models.

Second, for computational reasons we only performed 2-fold CV (k=2, random shuffle) during each epoch. This could have been easily changed for future runs to get much higher accuracies.

Third, more annotated data and epochs are needed for each model and essential for predictive deep learning in general.

There is a possible correlation between heart & lung diseases [4] , and this may cause unanticipated biases in our results (especially age cross-correlations). This deserves further investigation and study.

Conclusions

We have shown that even with limited data, deep learning models can predict some forms of lung cancer only by seeing the structure and intensity of mass within the cardiovascular regions of the heart. This confirms our hypothesis that lung and heart disease are significantly correlated, specifically that Coronal Artery Calcium (CAC) within arteries is also positively correlated with lung diseases, and their propensity to be benign, malignant, or malignant metastatic (spreading).

There needs to be a lot more work to scientifically justify our claims, including the following tasks: 1) Run standard CAC risk score algorithms, including patient age, race, etc., and apply an Agatson risk score 2) Run our ResNet 3D CNN architecture with proper "ground truth" images of CAC in arteries 3) Ensure overbalancing is corrected to 50/50 by arguing the strong correlation of lung/heart diseases and CAC present makes them part of the higher risk class 4) Apply a similarity score to measure the accuracy of a deep learning CAC risk score with both rule-based CAC (counting pixels matching criterion) and ML-based CAC (ML on the MESA dataset we first showed)

CAC threshold ranges we would like to incorporate into our method:

- Lipid-rich plaque: 47+-29 HU (range 18-76)

- Fibrous (calcified) plaque: 86+-29 HU (range 57-115)

CT's with or without lumen-enhancing contrast show no statistical significance on these values, so both were used in our dataset [5].

References

[1] S. Trajanovski *et al.*, "Towards radiologist-level cancer risk assessment in CT lung screening using deep learning," pp. 1–11, 2018.

[2] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, vol. 2016-Decem, pp. 770–778.

[3] S. G. Armato *et al.*, "The Lung Image Database Consortium (LIDC) and Image Database Resource Initiative (IDRI): A completed reference database of lung nodules on CT scans," *Med. Phys.*, vol. 38, no. 2, pp. 915–931, 2011.

[4] Harvard Health Letter, "When you look for cancer, you might find heart disease - Harvard Health." [Online]. Available: https://www.health.harvard.edu/heart-health/when-you-look-for-cancer-you-might-find-heart-disease. [Accessed: 09-Dec-2019].

[5] W. Kristanto, P. M. A. van Ooijen, M. C. Jansen-van der Weide, R. Vliegenthart, and M. Oudkerk, "A Meta Analysis and Hierarchical Classification of HU-Based Atherosclerotic Plaque Characterization Criteria," *PLoS One*, vol. 8, no. 9, pp. 1–13, 2013.