## Materials and Methods

### Architectures

We investigated several alterations on convolutional neural networks for feature extraction, into densely connected networks for outcome. The framework of the feature extraction has three parts: First, convolution blocks are performed in a manner similar to the DenseNet(Huang et al., n.d.) architecture, where previous feature maps are concatenated immediately to later layers. Second, a transition block reduces the number of filters by a fraction ranging from 0.5 to 1, where 1 indicates no reduction and 3D max pooling with a stride of 2. Third, feature extraction ends with a global max pooling layer.

The framework of the densely connected network has two parts: First, a set number of dense connections followed by dropout layer. Second, the model predicts two classes, progression and no-progression, with a soft-max activation.

We investigated several parameters in both the feature extractor, densely connected prediction, and parameters outside of the architecture.

Feature extraction parameters:

* 4, 8, 16, or 32 filters as the number of filters to start the feature extraction
* 4, 8, 16, or 32 as the rate of increase in the number of filters (growth rate)
* 1, 2, 3, 4, or 5 convolution blocks in each dense layers
* 1, 2, 3, or 5 dense convolution blocks followed by transition layers
* 0.5, 0.75, or 1.0 the fraction of filters to decrease by each transition

Densely connected parameters:

* 0, 1, 2, 3, 4, 5, or 7 number of densely connected layer
* 64, 128, 256, or 512 number of connections in each layer
* 0.5 or 0.0 dropout for fraction of connections to drop

Other hyper-parameters:

* Min and max learning rate were created on a model by model basis
* Adam or Stochastic gradient descent optimizers
* Categorical cross entropy, or Cosine loss

### Data

#### Preparation

We retrospectively acquired 119 patients who’d undergone percutaneous ablation therapy (PTA) for metastatic colorectal liver metastasis (CRLM) at our institution. Patient selection criteria included: a minimum follow-up period of 6 months, and a maximum of 2 months between pre-treatment and post-treatment contrast enhanced CT imaging. Follow-up data included three possible outcomes: residual disease, local progression, and no progression. For this work, labels 1 and 2 were combined as progression.

CRLM were manually segmented on pre-treatment contrast-enhanced CT scans, and validated by a radiology trained physician fellow. Segmentations were labeled based on the outcome status of the individual site, ‘GTV\_Recurred’ for progression and ‘GTV\_Not\_Recurred’ for no progression. Each disease site was individually labeled, resulting in a total of 197 sites.

An example of a patient with three sites, two of which locally progressed, Figure 1.

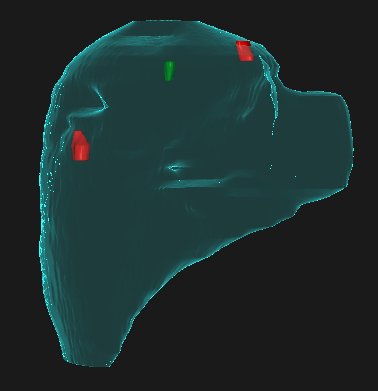


Figure : Example of patient with three disease sites. Two of the treated sites developed progression (red) and one site did not progress (green). Note that the expansion of the disease site outside of the liver is an artifact of 3D rendering.

Note that the expansion of the disease site outside of the liver is an artifact of 3D rendering, all sites are kept within the liver using ROI algebra.

#### Image Preparation

Automatic intensity based image registration was performed focused about the liver within the Raystation Treatment Planning system(Bodensteiner, 2018). Liver contours were generated automatically with our in-house liver auto-contouring algorithm(Anderson et al., 2020), manually inspected, and edited as needed.

Following rigid-registration, biomechanical deformable image registration within Raystation (Morfeus(Cazoulat et al., 2016)) was applied using the liver contours as boundary conditions. The resampled, post-ablation image, pre-ablation image, and unsampled post-ablation image were then exported for presentation to the model.

Images and segmentations were converted into nifti files using our in-house DicomRTTool (Anderson et al., 2021). Using SimpleITK(Beare et al., 2018), all images and masks are resampled to dimensions of 5x1x1mm, with bilinear interpolation and nearest-neighbor resampling, respectively.

#### Presentation to model

Image ‘slabs’ of size 32x64x64 were extracted for each treated site, centered about the segmented disease. Several examples of this, with the pre-treatment image on the left, and the post-ablation on the right, are shown in Figure 2.

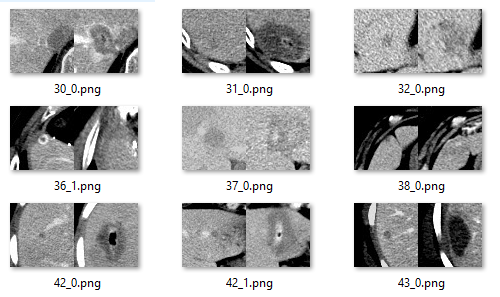


Figure : Examples of ‘slabs’ pulled form a variety of patients. Image on the left is pre-treatment and image on the right is post-ablation. Note that the nomenclature is patient ID\_site#, so 42\_0 and 42\_1 are both from the same patient.

#### Distributing the data

Data was randomly shuffled based on patient ID, placing each patient in the validation group until 1/3rd of the progression sites were in the validation set. Next, the remaining patients were moved into validation until 1/3rd of the non-progression sites were in the validation set. All remaining patients were placed in the training set. **Note that by splitting on patient ID we ensure no patients are distributed across both training and validation sets.** The final distribution of data can be seen in Table 1.

Table : Distribution of patients and lesions between train and validation set. Note that patients were split up based on patient ID, so a single patient will never be in both training and validation groups.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Source Information | | | Status | Image Distribution | |
| Source | Patients | Lesions | ~Progression | Train | Validation |
| Progression |
| MD Anderson  Cancer Center | 119 | 197 | 132 | 93 | 39 |
| 65 | 43 | 22 |

### Training the model

We investigated presenting the data in eight different ways, involving combinations of the pre-treatment image, the rigidly registered post-treatment image, the deformably registered post-treatment image, segmentations of the liver, and disease. A breakdown the combinations is shown in Table 2.

Table : The methods of presenting data to the model. Note that across all 8 the pre-treatment contrast enhanced CT is presented. Next, either the deformed or rigidly registered post-treatment image is presented. Segmentations of the liver and or disease are also included.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Imaging | | | Contours | |
| Pre-Treatment | Post-Treatment Deformed | Post-Treatment Rigid | Liver | Disease |
| x | x |  |  |  |
| x | x |  | x |  |
| x | x |  |  | x |
| x | x |  | x | x |
| x |  | x |  |  |
| x |  | x | x |  |
| x |  | x |  | x |
| x |  | x | x | x |

To balance the presentation of progression and no-progression, we created two TensorFlow data-generators with interleaving outputs. With a batch size of 32, this ensured each training step had 16 sites with progression and 16 sites without progression. Each generator will reshuffle and repeat, meaning the progression data is sampled more often than the no-progression data. This decision was influenced by recommendations from Tensorflow when working with imbalanced data, <https://www.tensorflow.org/tutorials/structured_data/imbalanced_data>.

#### Cosine Loss

Due to the limited, and biased (more no-progression than progression) nature of our dataset, we quickly noticed a problem of model overfitting. Normalization was introduced to the model in the form of l2 normalization and dropout in the dense connections, unfortunately it did not seem to alleviate the overfitting.

Recent work has shown that a large part of this issue can be related to the usage of softmax + crossentropy loss function(Barz & Denzler, n.d.), and propose the usage of a cosine loss function. The cosine loss focuses on increasing the similarity between the L2-norm of the prediction and the ground truth, ‘bounding’ the loss to a unit sphere. In contrast, the cross entropy and softmax contains exponential and log functions, allowing arbitrarily high values to appear.

After adopting the cosine loss function the model appeared to have less overfitting, and was adopted for the rest of the work.

#### Learning Rates

Unique minimum and maximum learning rates were identified using our inhouse learning rate finder (GitHub link redacted), adapted from code presented in <https://www.pyimagesearch.com/2019/08/05/keras-learning-rate-finder/>. These minimum and maximum learning rates were used in a Cosine annealing learning rate scheduler (GitHub link redacted). Step-size was varied to be 2000, 5000, or 10,000, with a reduction factor of 0.5.

### Evaluating Training

Training was monitored using the Tensorboard module(Abadi et al., n.d.). Area under the curve (AUC) was the metric of choice for overall model performance.