SUPPLEMENTARY METHODS

Deep Profiler and Multi-task Learning

All images were resampled to a 1.17 mm x 1.17 mm x 3 mm resolution. The original Hounsfield Unit (HU) values were clipped at -1,000 and 1000 and rescaled to -1 to 1 to better fit the input range of our model. To reduce complexity, we delimited a local region of interest that encompassed the GTV. The original CT volume for each patient was cropped to a 64 x 64 x 32 sub-volume or ~7.5 cm x 7.5 cm x 9 cm, chosen to encompass all tumors. Although the original CT volume could be used as input, given the limitation of training sample size, we selected sub-volumes to add prior information, mitigate over-fitting and avoid spurious associations at a distance from the tumor. The GTV mask was used as an additional channel rather than masking the CT volume, permitting flexibility for the network to learn features from both inside and outside of the GTV. Given that the image features extracted by the network is expected to be minimally influenced by the bounding box position (i.e. shift invariant) as long as the box encompasses the GTV, we randomly selected the cropped regions for each iteration of the training process. This data augmentation technique facilitated the network's ability to locate the desired invariant. Moreover, since the data provided to the neural network and classical radiomics, namely the CT volume and GTV mask, were similar, a more equitable comparison could be made between the two methodologies.

Deep Profiler consists of an encoder for extracting imaging features and building task-specific fingerprint, a decoder for estimating handcrafted radiomic features, and a task-specific network for generating image signature for therapy outcome prediction. A 3-D convolutional neural network (CNN) was used as an encoder for extracting imaging features. CNNs have been successfully applied to the field of image segmentation, classification and object detection.[1] The input layer to the encoder network included two channels, a CT image and GTV contour. The encoder consisted of 8 convolutional layers of 3 x 3 x 3 filter size. There were 16 filters in the first layer with the filter size doubled every two layers. There were max pooling operations after the second, fourth and sixth convolutional layers, with a pooling size of 2 x 2 x 2. The last convolutional layer was followed by an average pooling layer for extracting one feature from each feature map. Therefore, the encoder network reduced the feature dimension from 64 x 64 x 32 (input volume) to 128 (latent space). In all convolutional layers, we employed rectified linear units (ReLUs) as the nonlinear activation function. Batch normalization was applied after each convolutional layer.

Given that handcrafted radiomics are relatively efficient yet generic representations of tumor phenotype (*e.g.* size, shape, texture, etc.), we designed a decoder network to reconstruct radiomic features from latent fingerprint variables. Since the decoder network is tasked to reconstruct a near full set of radiomics features, we posited that the entire network would be more constrained and regularized, and thus less susceptible to overfitting. A fully-connected layer was adopted to link the latent fingerprint space with handcrafted radiomics. The loss function in this case is defined as mean square error between the decoder output and handcrafted radiomic feature values. Since radiomic features have different physical meanings and units, all radiomic features were normalized before they were used for training the network.

We used another neural network to predict the outcome based on latent fingerprint variables. Deep neural network models are typically used to solve classification problems. In the clinical setting, however, therapeutic outcomes are not merely binary but also contain time-to-event information. The Cox proportional hazards model, which assumes the risk of failure to be a linear combination of all covariates, is commonly used to relate the time that passes before local failure occurs. However, in many conditions it might be too simplistic to assume that the risk function is in a linear form. Recently, a fully connected network that models nonlinear associations between covariates was shown to outperform standard linear Cox regression.[2] In this study, we used the deep neural network to build a signature from the input image. A CNN-based encoder network was used to extract image features and a fully-connected layer was adopted to combine all image features into one Deep Profiler signature. This signature was made equivalent to the logarithm of the hazard ratio in the Cox regression, which is a linear combination of all input variables. Similar to the Cox model, there were no assumptions made about the form of the baseline hazard function except that the hazard ratio for an individual remains constant over time.

To fit the model, the coefficient is estimated using the maximum likelihood method in much the same way as it is done with Cox regression. Let $\mathbf{X}_i$ denote the input image of patient i with failure (or censoring) time t_i and $\varphi_\theta(\cdot)$ the network model with parameters θ. Treating patients as if they were statistically independent of each other, we computed the survival loss function of the network as a negative *log* partial likelihood of all samples in the training dataset:

$$\mathcal{L}_s(\theta) = -\log \prod_{i=1}^{N} \left[\frac{\exp\{\varphi_\theta(\mathbf{X}_i)\}}{\sum_{j=1}^{N} Y_{ij} \exp\{\varphi_\theta(\mathbf{X}_i)\}} \right]^{\delta_i},$$

where $Y_{ij} = 1$ if $t_j \geq t_i$, and $Y_{ij} = 0$ if $t_j < t_i$. Of note, we added an indicator for data censoring where $\delta_i = 1$ if event and $\delta_i = 0$ if censored.

Multi-task Learning

We used a multi-task learning framework to add constraints and regularity to the neural network so that the fingerprint can reconstruct radiomics features and accurately estimate the time to event for each patient. Multi-task learning has been used successfully across all applications of machine learning from natural language processing to computer vision. In our study, two distinct tasks were trained simultaneously **(Figure 1b**). The main task is the outcome prediction, which has the loss function in the form of $\mathcal{L}_s$. The auxiliary task is the estimation (reconstruction) of handcrafted radiomics, which is formulated as a mean square loss function $\mathcal{L}_r$:

$$\mathcal{L}_r(\theta) = \frac{1}{N}\sum_{i=1}^{N}\|\mathbf{O}_i - \mathbf{R}_i\|_2^2,$$

where N is the total number of patients, $\mathbf{O}_i$ is the decoder network output of patient i and $\mathbf{R}_i = \left(r_i^{(1)}, r_i^{(2)}, \cdots, r_i^{(k)}\right)^{\mathrm{T}}$ is a vector of k distinct handcrafted radiomic features of patient i. The total loss is defined as a weighted combination of two tasks:

$$\mathcal{L}_{\text{total}} = \lambda_s \mathcal{L}_s + \lambda_r \mathcal{L}_r$$

Each task loss is weighted by a scalar quantity to model the importance of each task on the combined loss $\mathcal{L}_{\text{total}}$. The weighting terms λ_s and λ_r in the multi-task loss function introduced additional hyper-parameters. The ratio between these two hyper parameters determined how much handcrafted information was kept for the outcome prediction. The values of these hyper-parameters could be equated or optimized through a grid-search in the training set. We applied $\boldsymbol{L}_2$ regularization on all network parameters (weight decay of 1×10^{-4}) and used Adam optimization with learning rate of 1×10^{-5}. Our model implementation is based on PyTorch (http://pytorch.org).

REFERENCES

1. Litjens G, Kooi T, Bejnordi BE, et al. A survey on deep learning in medical image analysis. *Med Image Anal* 2017; **42**: 60-88.
2. Katzman JL, Shaham U, Cloninger A, Bates J, Jiang TT, Kluger Y. DeepSurv: personalized treatment recommender system using a Cox proportional hazards deep neural network. *Bmc Med Res Methodol* 2018; **18**.

Implementing Deep Profiler

Algorithm 1 Train Deep Profiler Model

Input:

$Image$	preprocessed 3D CT images for training and validation
GTV	binary 3D GTV masks
$CensorFlag$	indicator of censoring, 1 if event and 0 if censored
$SurvivalTime$	time of event or censoring
$Radiomics$	hand-crafted radiomic features

Output:

$DP\hat{score}$	deep profiler score

```
 1: procedure TrainModel($Image$, $GTV$, $CensorFlag$, $SurvivalTime$, $Radiomics$)
 2:     $minPLloss \leftarrow +\infty$    ▷ minimum partial likelihood loss of failure prediction
 3:     $minTOTALloss \leftarrow +\infty$    ▷ minimum total multi-task loss
 4:     $\alpha \leftarrow 1 \times 10^{-5}$    ▷ set learning rate
 5:     $\eta \leftarrow 1 \times 10^{-4}$    ▷ set weight for L2 regularization
 6:     $\lambda_1, \lambda_2 \leftarrow 1$    ▷ weights for different functions
 7:     $\Theta \leftarrow$ XavierNormal($\sqrt{2}$)    ▷ model weights initialization
 8:     $training, validation \leftarrow$ RandomSplit(PatientIndex, 0.8, 0.2)
 9:     while $epoch <$ MaxEpoch do
10:         set the DeepProfiler network in training mode
11:         $DP\hat{score}, Radi\hat{o}mics \leftarrow$ DeepProfiler($Image[training], GTV[training]; \Theta$)
12:         $PLloss \leftarrow$ PredictionLoss($DP\hat{score}, CensorFlag[training], SurvivalTime[training]$)
13:         $MSEloss \leftarrow$ ReconLoss($Radi\hat{o}mics, Radiomics[training]$)
14:         $TOTALloss \leftarrow \lambda_1 PLloss + \lambda_2 MSEloss + \frac{\eta}{N}\sum_{\Theta}\|\Theta_i\|^2$
15:         $\Delta\Theta \leftarrow \nabla_{\Theta} TOTALloss$    ▷ Use backpropagation to compute the gradient
16:         $\Theta \leftarrow \Theta - \alpha \cdot \Delta\Theta$    ▷ Update the model parameters

17:         set the DeepProfiler network in evaluation mode
18:         $DP\hat{score}, Radi\hat{o}mics \leftarrow$ DeepProfiler($Image[validation], GTV[validation]; \Theta$)
19:         $PLloss \leftarrow$ PredictionLoss($DP\hat{score}, CensorFlag[validation], SurvivalTime[validation]$)
20:         $MSEloss \leftarrow$ ReconLoss($Radi\hat{o}mics, Radiomics[validation]$)
21:         $TOTALloss \leftarrow \lambda_1 PLloss + \lambda_2 MSEloss$
22:         if $TOTALloss < minTOTALloss$ then
23:             $\Theta_{\text{minTOTALloss}} \leftarrow \Theta$
24:             $minTOTALloss \leftarrow TOTALloss$
25:         end if
26:         if $PLloss < minPLloss$ then
27:             $\Theta_{\text{minPLloss}} \leftarrow \Theta$
28:             $minPLloss \leftarrow PLloss$
29:         end if
30:         $epoch \leftarrow epoch + 1$
31:     end while
32:     return $\Theta_{\text{minPLloss}}$, $\Theta_{\text{minTOTALloss}}$
33: end procedure
```

Algorithm 2 Calculate Loss Functions

```
Input:
    $DPscore$            deep profiler scores
    $CensorFlag$         indicator of censoring, 1 if event and 0 if censored
    $SurvivalTime$       time of event or censoring
Output:
    $nLogPL$             prediction loss, negative log of partial likelihood
 1: function PredictionLoss($DPscore$, $CensorFlag$, $SurvivalTime$)
 2:     $N \leftarrow$ Length($DPscore$)                              ▷ number of patients
 3:     $PL \leftarrow 1$
 4:     for $i = 1 \rightarrow N$ do
 5:         $T[i] \leftarrow 0$
 6:         for $j = 1 \rightarrow N$ do
 7:             if $SurvivalTime[i] \leq SurvivalTime[j]$ then
 8:                 $Y \leftarrow 1$
 9:             else
10:                 $Y \leftarrow 0$
11:             end if
12:             $T[i] \leftarrow T[i] + Y \cdot \exp(DPscore[j])$
13:         end for
14:         $L[i] \leftarrow \exp(DPscore[i])/T[i]$
15:         $PL \leftarrow PL \cdot$ Power($L[i], CensorFlag[i]$)
16:     end for
17:     $nLogPL \leftarrow -\log PL$            ▷ use negative log of the likelihood as object fucntion
18:     return $nLogPL$
19: end function

Input:
    $\hat{Radiomics}$    estimation of radiomic features
    $Radiomics$          hand-crafted radiomic features
Output:
    $MSE$                reconstruction loss, mean squared error between feature vectors
20: function ReconLoss($\hat{Radiomics}$, $Radiomics$)
21:     $N_p \leftarrow$ Size($Radiomics, 1$)                          ▷ number of patients
22:     $N_f \leftarrow$ Size($Radiomics, 2$)                          ▷ number of features
23:     $Error \leftarrow 0$
24:     for $i = 1 \rightarrow N_p$ do
25:         for $j = 1 \rightarrow N_f$ do
26:             $Error \leftarrow Error + |\hat{Radiomics}[i, j] - Radiomics[i, j]|^2$
27:         end for
28:     end for
29:     $MSE \leftarrow \frac{1}{N_p N_f} Error$
30:     return $MSE$
31: end function
```

Algorithm 3 Calculate Deep Profiler Score

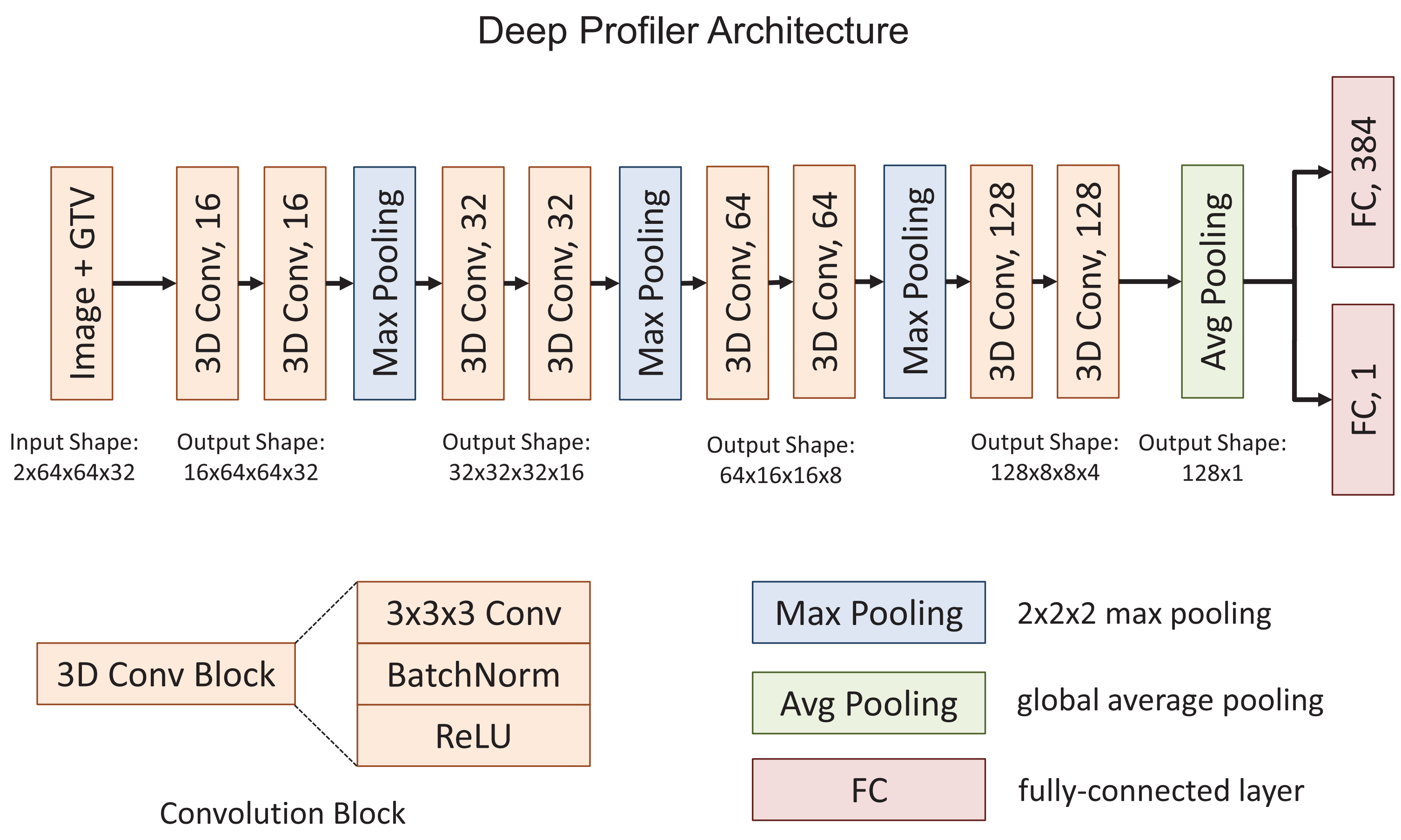
```
Input:
    $TestingImage$        preprocessed 3D CT image for testing
    $TestingGTV$          binary 3D GTV mask
    $\Theta_{\text{minPLloss}}$          parameters of deep profiler model with minimum validation loss
Output:
    $\hat{DPscore}$            deep profiler score
 1: function TESTMODEL($TestingImage$, $TestingGTV$, $\Theta_{\text{minPLloss}}$)
 2:     set the DeepProfiler network in evaluation mode
 3:     $\hat{DPscore}, \hat{Radiomics} \leftarrow$ DEEPPROFILER($TestingImage, TestingGTV; \Theta_{\text{minPLloss}}$)
 4:     return $\hat{DPscore}$
 5: end function
```

Algorithm 4 Calculate Saliency Map

```
Input:
    $TestingImage$        preprocessed 3D CT image for testing
    $TestingGTV$          binary 3D GTV mask
    $\Theta_{\text{minPLloss}}$          parameters of deep profiler model with minimum validation loss
Output:
    $SaliencyMap$         saliency map of each input volume
 1: function GETSALIENCY($TestingImage$, $TestingGTV$, $\Theta_{\text{minPLloss}}$)
 2:     set the DeepProfiler network in evaluation mode
 3:     for $i = 1 \rightarrow$ LENGTH($TestingImage$) do                    ▷ number of patients
 4:         $\hat{DPscore}, \hat{Radiomics} \leftarrow$ DEEPPROFILER($TestingImage[i], TestingGTV[i]; \Theta_{\text{minPLloss}}$)
 5:         $SaliencyMap[i] \leftarrow \nabla_{TestingImage[i]} \hat{DPscore}$
 6:     end for
 7:     return $SaliencyMap$
 8: end function
```

Algorithm 5 Pre-process CT image, GTV mask and Radiomics

```
Input:
    $RawImage$            raw 3D CT image
    $RawGTV$              raw 3D GTV mask
Output:
    $Image$               preprocessed CT image
    $GTV$                 preprocessed GTV mask
    $Radiomics$           normalized radiomic features
 1: function PREPROCESSING($RawImage$, $RawGTV$)
 2:     $N \leftarrow$ LENGTH($RawImage$)                                        ▷ number of patients
 3:     $VoxelSpacing \leftarrow [1.17, 1.17, 3]$
 4:     $xSize, ySize, zSize \leftarrow [64, 64, 32]$
 5:     for $i = 1 \rightarrow N$ do
 6:         if GETVOXELSPACING($RawImage[i]$) $\neq VoxelSpacing$ then
 7:             $RawImage[i] \leftarrow$ RESAMPLE($RawImage[i], VoxelSpacing, order = 1$)        ▷ linear interpolation
 8:             $RawGTV[i] \leftarrow$ RESAMPLE($RawGTV[i], VoxelSpacing, order = 0$) ▷ nearest neighbour interpolation
 9:         end if
10:         $Cx, Cy, Cz \leftarrow$ CENTEROFMASS($RawGTV[i]$)
11:         $GTV[i] \leftarrow RawGTV[i, Cx \pm xSize/2, Cy \pm ySize/2, Cz \pm zSize/2]$
12:         $Image[i] \leftarrow RawImage[i, Cx \pm xSize/2, Cy \pm ySize/2, Cz \pm zSize/2]$
13:         $Image[i] \leftarrow Image[i]/1000$
14:         $Radiomics[i] \leftarrow$ RADIOMICSENGINE($RawImage[i], RawGTV[i]$)
15:     end for
16:     $Radiomics \leftarrow$ ZSCORE($Radiomics$)
17:     return $Image$, $GTV$, $Radiomics$
18: end function
```


Deep Profiler Architecture
Image + GTV
3D Conv, 16
3D Conv, 16
Max Pooling
3D Conv, 32
3D Conv, 32
Max Pooling
3D Conv, 64
3D Conv, 64
Max Pooling
3D Conv, 128
3D Conv, 128
Avg Pooling
FC, 384
FC, 1
Input Shape: 2x64x64x32
Output Shape: 16x64x64x32
Output Shape: 32x32x32x16
Output Shape: 64x16x16x8
Output Shape: 128x8x8x4
Output Shape: 128x1
3D Conv Block
3x3x3 Conv
BatchNorm
ReLU
Convolution Block
Max Pooling
2x2x2 max pooling
Avg Pooling
global average pooling
FC
fully-connected layer