Supplementary Materials

Predicting the Outcome of Radiotherapy in Brain Metastasis by Fusing the Clinical Attributes and Deep Learning Features of MRI

Seyed Ali Jalalifar[(1)], Hany Soliman[(2,3,5)], Arjun Sahgal[(2,3,5)], and Ali Sadeghi-Naini[(1,2,4,5)]

(1) Department of Electrical Engineering and Computer Science, Lassonde School of Engineering, York University, Toronto, ON, Canada
(2) Department of Radiation Oncology, Odette Cancer Centre, Sunnybrook Health Sciences Centre, Toronto, ON, Canada
(3) Department of Radiation Oncology, University of Toronto, Toronto, ON, Canada
(4) Department of Medical Biophysics, University of Toronto, Toronto, ON, Canada
(5) Physical Sciences Platform, Sunnybrook Research Institute, Sunnybrook Health Sciences Centre, Toronto, ON, Canada

Components of the Proposed Deep Network Framework

The main components in the proposed framework consist of a slice-by-slice MRI feature extractor, a recurrent network or transformer, and a module to integrate clinical features to the deep-learning-based MRI features. The rationale for using each of these components is due to the nature of input data and the task at hand. The input data consists of two 3D MRI volumes (T1w and T2-FlLAIR) each with a size of 128×128×45 voxels (128×128×2×45 voxels in total.) One approach to represent and analyze this input is to interpret it as a series of 128×128×2 pixel spatially dependent images. This approach, while logical, also opens the door to utilize the widely used convolutional neural networks (CNN) as feature extractors. Whereas the 128×128×2 pixel slice pairs consist of 32,768 pixels (potential features), they could be reduced to 256 meaningful and distinctive features for each slice pair by training a CNN to classify treatment outcome of the tumor associated with that slice. For this purpose, we adapted an InceptionResNetV2 as the feature extractor, as it is a powerful model with proven performance on different benchmarks for images classification and feature extraction. At the end of the feature extraction step, 256 features are derived for each slice (a total of 256×45 features for each two-channel MRI input volume.) At this point, the 3D spatial dependency of these features in a volumetric MRI has not been taken into account yet. To do so a recurrent network or transformer was adapted as the deep learning architectures proposed for processing series. We also encoded and fused standard clinical features with the MRI deep learning features to investigate whether they can provide complementary information to the framework for therapy outcome prediction in brain metastasis. To summarize, the rationale for using InceptionResNetV2 is based on the fact that our input is two-channel MRI, the reason for including an LSTM, sequence to sequence, or transformer network is because our images are volumetric and spatially dependent, and the rationale for integrating clinical features is that they can potentially provide additional relevant information that is available before starting treatment but may not be possible to extract from MRI.

1. InceptionResNetV2

Residual connections and inception blocks have been central in many advances in computer vision in recent years. The InceptionResNetV2 borrowed the idea from both and proposed combining inception architecture with residual connections. Residual connections were first introduced by He *et. al.* to ease the training of very deep networks [1]. A basic residual block is demonstrated in

Figure S1(a). In the deep residual learning regime, instead of stacking non-linear layers to find the desired underlying mapping $\mathcal{H}(x)$, nonlinear layers are stacked to fit a residual mapping $\mathrm{F}(x) \coloneqq \mathcal{H}(x) - x$. Theoretically, it is easier to optimize the residual mapping rather than original mapping. In the extreme case, if the optimal solution is identity mapping, it would be easier to push $\mathrm{F}(x)$ to zero rather than learning identity mapping by stacking nonlinear layers [1].

The idea of the inception module is to make networks wider rather than deeper by having filters of multiple size operating on the same level instead of stacking them [2]. Extra 1×1 convolutions were added in the developed framework to limit the number of output channels by dimensionality reduction (*Figure S1(b)*).

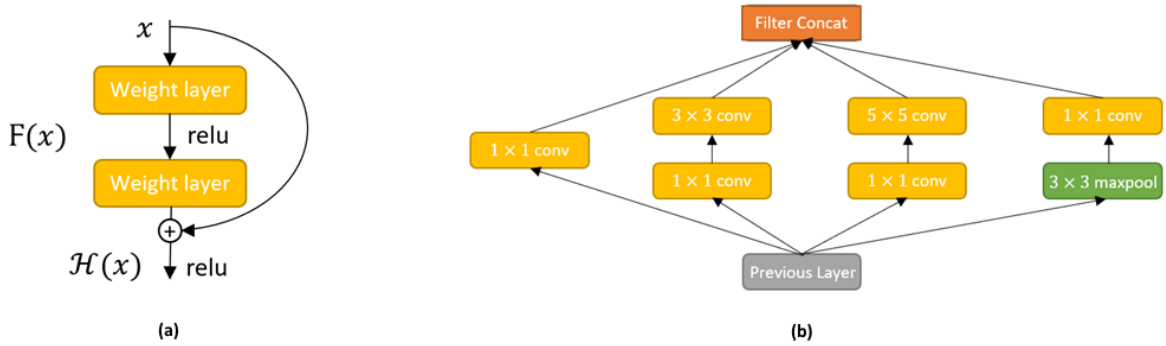

Figure S1. **(a)** A residual building block, and **(b)** Inception module with dimension reduction.

In InceptionResNetV2, residual connections allow for the network to have more layers while inception blocks make the networks wider [3]. *Figure S2* depicts the compressed view of the InceptionResnetV2 network adapted in the developed framework. Notice the residual connection replaced the concatenation filter in the last layer of inception blocks.

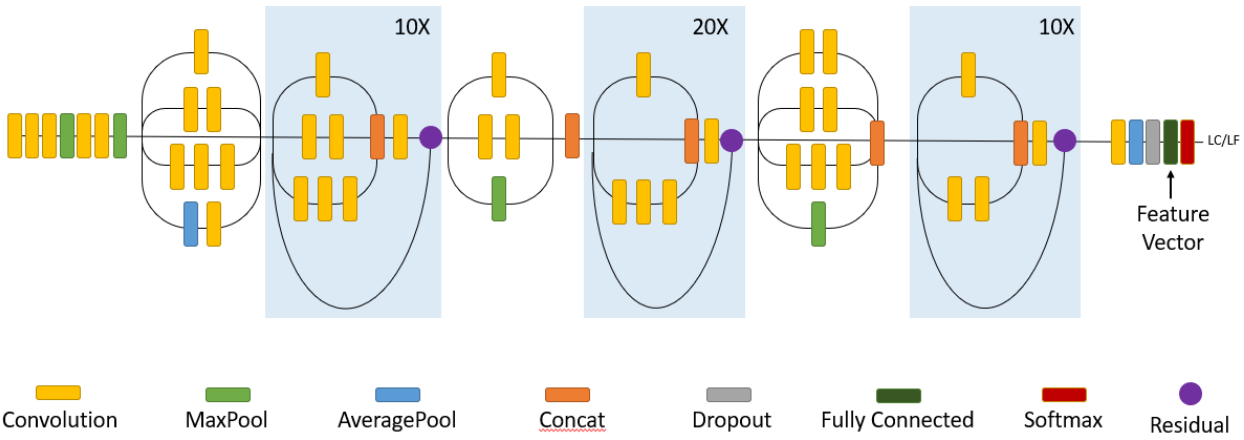

Figure S2. A compressed view of InceptionResNetV2. X shows number of times an Inception block with residual connection is repeated.

2. Recurrent Networks

Recurrent neural networks (RNN) allow information to persist over time through feedback connections [4]. LSTM is a special kind of recurrent neural network powerful at learning long-term dependencies. By having an internal mechanism called gate, LSTM can regulate the flow of information. These gates can learn which data in a sequence is important to keep or discard. By doing that, it can pass relevant information down the long chain of sequences to make predictions [4]. *Figure S3(a)* demonstrates inside of an LSTM cell. The forget gate decides whether the information should be kept, or it is redundant and should be discarded. In the input gate, the cell state is updated based on the multiplication of *tanh* output and *sigmoid* output. The output gate decides what the next hidden state should be. These gates together control the flow of information from one cell state to another and help with maintaining long-range dependencies.

The sequence to sequence (Seq2Seq) models [5] are a special kind of recurrent neural networks and are usually utilized to solve natural language processing problems such as machine translation, image captioning, question answering, etc. Most common Seq2Seq architectures consist of an encoder and a decoder. Both the encoder and decoder are LSTM models. Encoder reads the input sequence and summarizes the information into a context vector. The context vector is then fed to the decoder and the decoder tries to make accurate predictions based on the context vector. *Figure S3(b)* depicts the system overview of a Seq2Seq model. Instead of relying solely on the hidden state of the last LSTM cell as the context vector, the linear sum of the hidden states from all LSTM

cells in the encoder could be calculated as a context vector that brings about a seq2seq model with attention [6].

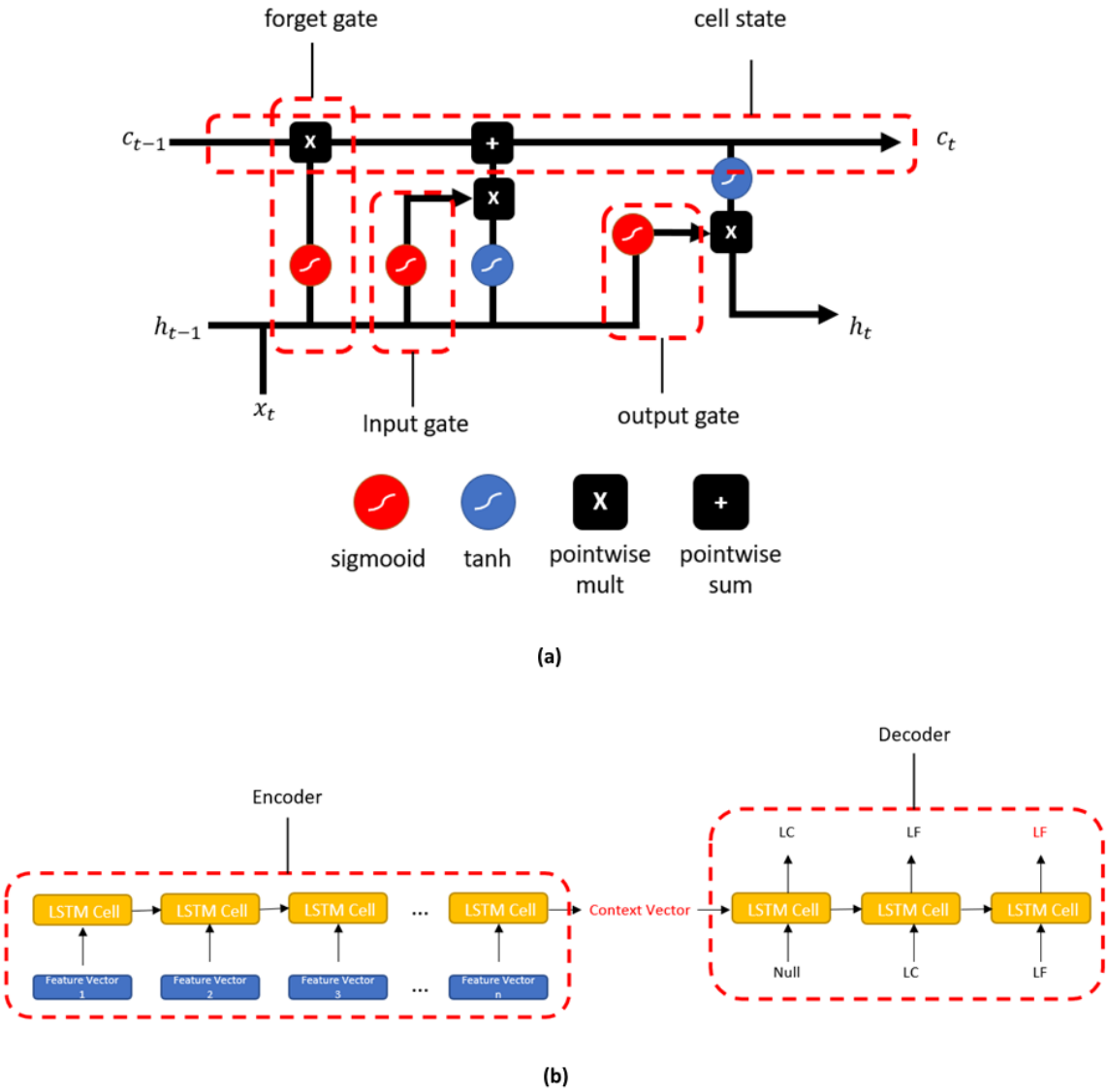

Figure S3. **(a)** Inside of an LSTM cell with *input gate*, *output gate*, *forget gate*, and *cell state*. $h_{t-1}, c_{t-1}, x_t. h_t, c_t$ are hidden state of the previous layer, cell state of the previous layer, input, hidden state of the current layer, and cell state of the current layer, respectively, and **(b)** System overview of a Seq2Seq model with encoder and decoder.

3. Transformers

A transformer is a novel architecture that obviates the need for recurrence in processing series and relies entirely on an attention mechanism to find global dependencies between input and output. Initially introduced by Vaswani et al. [7], transformer networks achieved state-of-the-art performance in sequence-to-sequence tasks compared to recurrent networks while being more

parallelized and requiring significantly less time to train. Similar to the Seq2Seq network, the transformer architecture employs an encoder-decoder structure, however, eliminating recurrence and replacing it with attention mechanisms allows for substantially greater parallelization than RNNs and CNNs. The transformer network was initially introduced as a model for sequence-to-sequence translation but since then, a large variety of networks have been introduced based solely on the encoder or decoder part of the transformer architecture. GPT-3 [8] is an example of a network that is built on the decoder part of the transformer that produces human-like text with very high quality. The encoder part of a transformer on the other hand has been utilized in architectures such as BERT [9] for question-answering and text classification tasks.

As described earlier, a transformer has an encoder-decoder architecture and uses a self-attention mechanism to infer dependencies. Each encoder or decoder consists of modules that contain feed-forward and attention layers. *Figure S4(a)* demonstrates a transformer architecture. With (key, value) vector pairings, the attention layer employs a trainable associative memory. From a series of N inputs, the Query and Key matrices are generated and packed into the following matrices:

$$X \in \mathbb{R}^{N\times D}; \quad Q = XW_Q \in \mathbb{R}^{N\times D}; \quad K = XW_K \in \mathbb{R}^{N\times D} \tag{1}$$

where X is the sequence of N inputs with dimension D, Q and K are the Query and Key matrices, and W_Q and W_K are linear transformations with trainable parameters The output of the attention is a weighted sum of the N-Value matrix $V \in \mathbb{R}^{N\times D}$ and is calculated by the following formula:

$$Attention(Q, K, V) = Softmax\left(\frac{QK^T}{\sqrt{d}}\right)V \tag{2}$$

Instead of offering a single attention head, Vaswani et al. [7] offered multi-head attention, which entails applying h self-attention functions to the input.

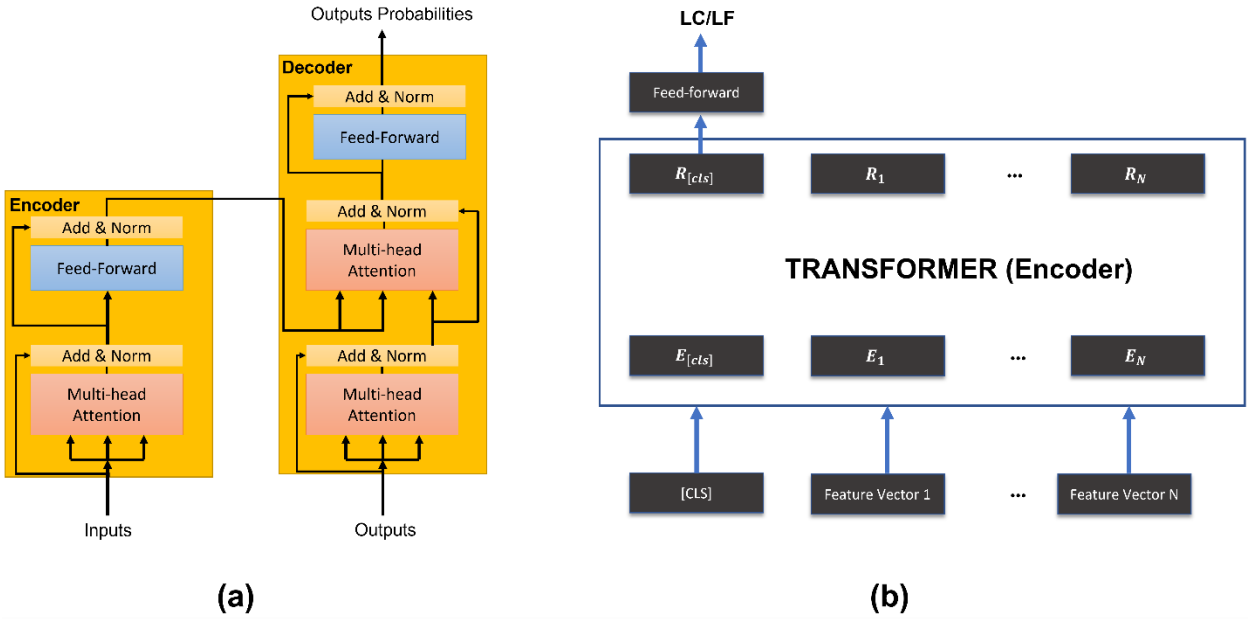

Figure S4. **(a)** The architecture of a transformer for sequence-to-sequence tasks. In practice, a stack of multiple encoders and decoders is used. Each encoder consists of multi-head attention later, residual connections, and feed-forward layer to prepare the input of next encoder/decoder block. **(b)** Classification using the encoder component of a transformer. A classification token [cls] is added to the input series and in the output of the encoder. $E_1, E_2, \dots, E_N$ are the input feature vectors with position encoding added to them. The input vectors along with $E_{[cls]}$ go through the encoder layers and output a sequence vector of $R_1, R_2, \dots, R_N$ along with $R_{[cls]}$. A feed-forward network attached to the $R_{[cls]}$ is used for final classification.

Experimental Results of Different Hyperparameters and Pre-training Settings

Table 1 – Experimental results of the proposed framework on Validation Set. For the input of LSTM, Seq2Seq, and Transformer networks, the extracted features from the best InceptionResNet were concatenated with clinical features and then fed to these networks.

Model	Pre-train (ImageNet)	Pretrain (ImageNet + BraTS)	Learning Rate	Batch Size	Epochs	Validation Accuracy	Validation Sensitivity	Validation Specificity
InceptionResNet	✓	✓	**1e-6**	**1**	**500**	**66.7%**	**66.7%**	**66.7%**
InceptionResNet	✓	×	1e-6	1	500	60%	66.7%	56%
InceptionResNet	×	×	1e-6	1	500	53.4%	50%	56%
InceptionResNet	✓	✓	1e-3	1	500	46.7%	50%	44.4%
InceptionResNet	✓	✓	1e-3	32	500	53.4%	50%	56%
InceptionResNet	✓	✓	1e-3	64	500	53.4%	50%	56%
InceptionResNet	✓	✓	1e-6	32	500	60%	66.7%	56%
InceptionResNet	✓	✓	1e-6	1	200	53.4%	50%	56%
LSTM	×	×	**1e-4**	**2**	**500**	**86.7%**	**83.3%**	**88.9%**
LSTM	×	×	1e-4	16	500	73.3%	66.7%	77.8%
LSTM	×	×	1e-2	2	500	73.3%	83.3%	66.7%
LSTM	×	×	1e-4	2	100	80%	66.7%	88.9%
Seq2Seq	×	×	**1e-4**	**2**	**500**	**80%**	**83.3%**	**77.8%**
Seq2Seq	×	×	1e-4	16	500	73.3%	83.3%	66.7%
Seq2Seq	×	×	1e-2	2	500	73.3%	66.7%	77.8%
Seq2Seq	×	×	1e-4	2	100	53.3%	33.3%	66.7%
Transformer	×	×	**1e-4**	**8**	**200**	**80%**	**83.3%**	**77.8%**
Transformer	×	×	1e-4	16	200	80%	66.67%	88.9%
Transformer	×	×	1e-2	8	200	73.3%	83.3%	66.7%
Transformer	×	×	1e-4	8	100	80%	66.67%	88.9%

References

[1] K. He, X. Zhang, S. Ren, and J. Sun, “Deep Residual Learning for Image Recognition,” Dec. 2015, [Online]. Available: http://arxiv.org/abs/1512.03385.

[2] C. Szegedy *et al.*, “Going Deeper with Convolutions,” Sep. 2014, [Online]. Available: http://arxiv.org/abs/1409.4842.

[3] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. Alemi, “Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning,” Feb. 2016, [Online]. Available: http://arxiv.org/abs/1602.07261.

[4] S. Hochreiter and J. Schmidhuber, “Long Short-Term Memory,” *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997, doi: 10.1162/neco.1997.9.8.1735.

[5] I. Sutskever, O. Vinyals, and Q. V. Le, “Sequence to Sequence Learning with Neural Networks,” Sep. 2014, [Online]. Available: http://arxiv.org/abs/1409.3215.

[6] D. Bahdanau, K. Cho, and Y. Bengio, “Neural Machine Translation by Jointly Learning to Align and Translate,” Sep. 2014, [Online]. Available: http://arxiv.org/abs/1409.0473.

[7] A. Vaswani *et al.*, “Attention is All you Need,” in *Advances in Neural Information Processing Systems*, 2017, vol. 30, [Online]. Available: https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

[8] T. B. Brown *et al.*, “Language Models are Few-Shot Learners,” May 2020, [Online]. Available: http://arxiv.org/abs/2005.14165.

[9] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding,” Oct. 2018, [Online]. Available: http://arxiv.org/abs/1810.04805.