ON-LINE APPENDIX

Overview of Convolutional Neural Networks

CNNs are an adaption of the traditional artificial neural network architecture whereby banks of 2D convolutional filter parameters and nonlinear activation functions act as a mapping function to transform a multidimensional input image into a desired output.[1] The banks of 2D convolutional operation are defined by

$$C_l = \sum_k x_k W_{kl} + b_l,$$

where the lth convolutional output C_l is the result of convolution between the kth input feature map x_k, and the lth subparameters w_{kl} and b_l are the lth additive bias terms. Because each convolutional layer acts on a total of k input feature maps, each parameter of size $i \times j$ consists of a total of $i \times j \times k \times l$ parameters, in which l is the number of output feature maps.

Each convolutional operation is then followed by a nonlinear activation function, σ. In this study, the rectified linear activation function was used, given well-documented advantages, including stable gradients at the extreme values of optimization.[2] The rectified linear operation is defined simply by

$$x_l = \sigma(C_l) = \max(C_l, 0),$$

where the lth activation map x_l represents the convolutional output C_l described above with the threshold at zero. Stacking serial convolutional and nonlinear activation functions allows a CNN to model high-order complex feature representations in a mathematically efficient form.

Convolutional Neural Network: Architectural Details

A customized CNN based on the popular ResNet was designed for classification of mutation status, comprising 4 residual blocks (On-line Figure). Batch normalization is used between the convolutional and rectified linear layers to limit drift of layer activations during training.[3] Dropout at 50% was applied to all convolutional and fully connected layers to limit overfitting and add stochasticity to the training process.[4,5] Feature maps were downsampled from the previous layer by convolutions with a stride length of 2 instead of max pooling, thus allowing the network to learn downsampling parameters and facilitating preservation of gradients during backpropagation.[6] The number of activation channels in deeper layers was progressively increased from 8 to 16 to 32 to 64, reflecting increased feature complexity. The final single-dimensional feature vector was obtained through use of a global average pool applied to the penultimate $4 \times 4 \times 64$ convolutional feature map instead of implementing a costly, high-parameter intermediate fully convolutional layer.

Final classification error was determined using a softmax cross-entropy loss function, defined by

$$y = -\sum_1 (x_{lc} - \log \sum_{d=1}^{D} e^{x_{ld}}),$$

where the loss, y, is calculated by subtracting the lth activation map of the ground truth class, c, with the sum of the softmax normalized (exponential function) values of the remaining class dimensions, D.

Data Augmentation

Real-time data augmentation was applied to all input images during the training process. This included the following: 1) addition of a random offset i on the interval of (−0.5 and 0.5) to the whole image; 2) arbitrary removal of an entire channel within the input in 50% of training cases; and 3) application of a random 3×3 affine transformation matrix independently to each input channel, resulting in stochastic application of image scaling, rotation, translation, and shear. "Channels" refer to 1 of the 4 input modalities (ie, T2, FLAIR, and so forth). Given a 2D affine matrix,

$$\begin{bmatrix} s_1 & t_1 & r_1 \\ t2 & s_2 & r_2 \\ 0 & 0 & 1 \end{bmatrix},$$

the random affine transformation was initialized with random uniform distributions of interval $s_1, s_2 \in (0.8, 1.2)$, $t_1, t_2 \in (-0.3, 0.3)$, and $r_1, r_2 \in (-16, 16)$.

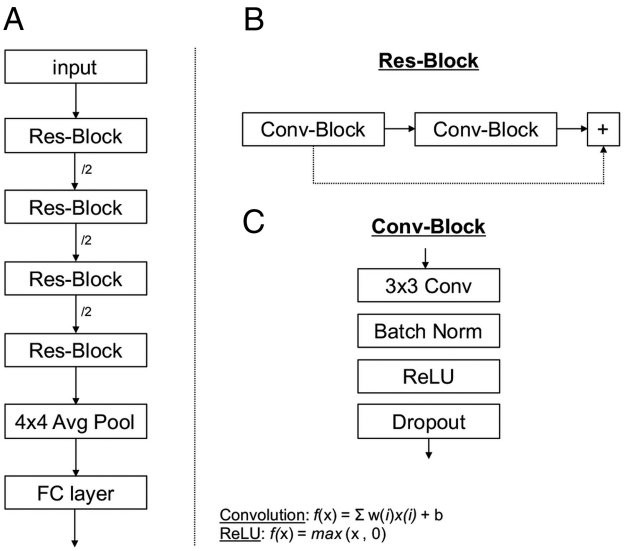
Implementation Details

Training was implemented using the Adam optimizer, an algorithm for first-order gradient-based optimization of stochastic objective functions, based on adaptive estimates of lower-order moments.[7] Parameters were initialized using the heuristic described by He et al.[8] L2 regularization was implemented to prevent over-fitting of data by limiting the squared magnitude of the convolutional weights. To account for training dynamics, the learning rate was annealed and the mini-batch size is increased whenever training loss plateaus. Furthermore a normalized gradient algorithm was employed to allow for locally adaptive learning rates that adjust according to changes in the input signal.[9] Software code for this study was written in Python 3.5 using the open-source TensorFlow r1.0 library (Apache 2.0 license).[10] Experiments were performed on a GPU-optimized workstation with a single NVIDIA GeForce GTX Titan X (12GB, Maxwell architecture).

REFERENCES

1. LeCun Y, Bengio Y. **Convolutional networks for images, speech, and time-series.** In: Arbib MA, ed. *The Handbook of Brain Theory and Neural Networks*. Cambridge: MIT Press; 1998:255–58
2. Nair V, Hinton GE. **Rectified linear units improve restricted Boltzmann machines**. In: *Proceedings of the 27th International Conference on Machine Learning*. Haifa, Israel: Omnipress; 2010:807–14
3. Ioffe S, Szegedy C. **Batch normalization: accelerating deep network training by reducing internal covariate shift.** *PMLR* 2015;37:448–56
4. Srivastava N, Hinton G, Krizhevsky A, et al. **Dropout: a simple way to prevent neural networks from overfitting.** *J Mach Learn Res* 2014; 15:1929–58
5. Baldi P, Sadowski P. **The dropout learning algorithm.** *Artif Intell* 2014;210:78–122 CrossRef Medline
6. Springenberg JT, Dosovitskiy A, Brox T, et al. **Striving for simplicity: the all convolutional net.** *CoRR* 2014;abs/1412.6806. https://arxiv.org/abs/1412.6806. Accessed August 30, 2017
7. Kingma DP, Ba J. **Adam: a method for stochastic optimization.** *CoRR* 2014;abs/1412.6980. https://arxiv.org/abs/1412.6980. Accessed August 30, 2017
8. He K, Zhang X, Ren S, et al. **Delving deep into rectifiers: surpassing human-level performance on ImageNet classification.** In: *Proceedings of the 2015 IEEE International Conference on Computer Vision*, Santiago, Chile. December 7–13, 2015:1026–34 CrossRef

9. Mandic DP. **A generalized normalized gradient descent algorithm.** *IEEE Signal Process Lett* 2004;11:115–18 CrossRef
10. Abadi M, Agarwal A, Barham P, et al. **TensorFlow: large-scale machine learning on heterogeneous distributed systems.** *CoRR* 2015; abs/1603.04467. https://arxiv.org/abs/1603.04467. Accessed August 30, 2017



ON-LINE FIGURE. *A*, Summary of residual neural network architecture. Four residual blocks are used, subsampling the feature map 3 times through convolutions with a stride of 2 (demarcated by /2 in the figure). *B*, Each residual block consists of 2 serial 3 × 3 convolutional blocks; the latter is mapped to the former via an additional operation. *C*, Each convolutional block consists of a 3 × 3 convolution, batch normalization, a rectified linear nonlinearity, and 50% drop-out.