10.1148/ryai.2019180015

Appendix E1

Hardware

All training and evaluation was performed on a standard desktop workstation with the following hardware: Intel i7 3770k 3.5 GHz Processor, 8 GB RAM, 250 GB Solid-State drive, and Nvidia GTX 1080 Ti GPU. We utilized the current version of the PyTorch deep learning framework (version 0.4.0) (36).

Report Classification

One of the great challenges facing the community of researchers creating deep learning models in radiology is the lack of well-annotated data. Despite the utilization of structured reporting at our institution, we similarly faced a challenge in classifying the largely heterogeneous bodies of text in various reports. We accordingly turned to machine learning to solve this problem and used the recently described fastText algorithm (11) to aid in rapidly classifying radiology reports into "normal" and "abnormal" studies.

We first manually classified 200 reports into normal and abnormal reports. The difference between normal and abnormal reports included subtle differences in phrasing such as, "indubitably, there is evidence of a joint effusion" and "I see there is no evidence of a joint effusion." We then trained a fastText classifier on the 200 reports and subsequently used the trained fastText classifier to assist in classifying an additional 200 reports. Assistance is defined here as the model "suggesting a categorization" as either normal or abnormal, with final choice ultimately being human. We then retrained the fastText model on the 400 manually classified reports and continued this process iteratively.

As the fastText model improved in accuracy, the human operator was allowed more opportunity to simply agree rather than have to override the suggested categorization, thus speeding up the process iteratively. In total, 2200 reports were classified manually with this method. The remaining 19256 studies were classified automatically with the algorithm. We chose 2200 as the stopping point based on randomly sampling 500 automatically classified reports (250 normal and 250 abnormal) and ensuring that all 500 reports were classified correctly. The use of such a natural language processing tool in this instance necessitated far less manpower to classify 21456 reports than would otherwise have been required.

Preprocessing

The DICOM source images were preprocessed into anonymized 8-bit PNG gray-scale images (Fig E1, Step 1), which were stored on the hard drive of our machine performing training. Every pixel of these 8-bit images was represented by an integer value in the range from 0 to 255. Each of these PNG images underwent a series of random transformations prior to being passed into the model throughout training (Fig E1, Steps 2–5); these transformations are easily defined in the PyTorch library and are further detailed in the Data Augmentation section below. The images are normalized into a 500 × 500 array of floating point numbers, where each pixel value is

represented in the range of 0.0 to 1.0. Thus, 8-bit integer values of 0, 127.5, and 255 are represented as 0.0, 0.5, and 1.0, respectively. The channel dimensionality of this 500 × 500 array is defined as 1 (single-channel) for gray-scale to distinguish it from a RGB color image with channel dimensionality of 3.

Data Augmentation

Data augmentation (Fig E1, Steps 2–5) is the process by which training set images are altered slightly each time they are seen throughout training to increase variability and effectively increase the number of training samples the model may see. For example, by utilizing a random rotation of 2 degrees, images may be rotated by −2, −1, 0 (no rotation), 1, or 2 degrees, prior to being passed into the model, ensuring the model sees slightly rotated variations of the same image. The following data augmentation was employed through both phases of training for our study: random rotation of 20 degrees, random resizing ranging between 510 × 510 pixels and 530 × 530 pixels, random crop of target image size (500 × 500), and random horizontal flips.

Softmax Function (Probabilistic Interpretation) (14)

$$P\left(y_i \mid x_i; W\right) = \frac{e^{f_{y_i}}}{\sum_j e^{f_j}},$$

where the probability of an output class y_i is normalized into the probabilities of each class based on the output vector f so that the probabilities sum to 1. As an example, if the output values are −1 and 1 for negative and positive, the softmax output would be 0.12 and 0.88, respectively. In Python, the code can be represented as:

$$\mathbf{Softmax} = \mathbf{lambda}\,\mathbf{x}: \mathbf{numpy.exp}\left(\mathbf{x}\right) \Big/ \mathbf{numpy.sum}\left(\mathbf{numpy.exp}\left(\mathbf{x}\right)\right),$$

where x is an array of the output values, with length of the array equal to 2 for our study (either positive or negative).

Rationale for CNN/Model Selection

The Inception architecture (37) has been shown in the literature to be versatile at a wide range of image classification tasks, including excellent classification of small bowel obstruction (5) and human-level detection of dermatologic conditions (38,39). We chose the Xception architecture (12), which is inspired by Inception but replaces the standard Inception modules with depth-wise separable convolutions, allowing for more efficient use of model parameters. We also favored the architecture for its relatively low parameter count (approximately 25 million parameters) compared to its performance on benchmark datasets such as ImageNet, where architectures on par with Inception-v3 have been shown to achieve human-level classification accuracy (14,40,41).

Most modern freely available published architectures in standard GPU accelerated deep learning frameworks, such as PyTorch (36), demonstrate excellent speed in inference, which was evidenced in our results with approximately 21 milliseconds to process a single image and 63 milliseconds to produce an output for an entire study. Speed of inference will be an important factor in the real-time application of classification in an acute care setting. As these frameworks continue to mature, we expect continued improvement in their performance and ease of use.

RNNs have enjoyed recent success in the problem domain of video processing, as suggested and cited in our Discussion section. One such method involves passing the CNN feature layer outputs to be processed as a whole by the RNN, analogous to the usage by our model in this article. A paper that explores this concept is by Sun et al (26). Simplistically, LSTMs and GRUs can be thought of as having "memory" maintained across timesteps. In the case of video processing, the RNN processes the extracted features from each frame in the video after it is passed through the trained CNN, achieving an overall decision about the video by using its "memory" to remember relevant items across multiple frames.

Another excellent resource for understanding RNNs is the CS231n course taught by Karpathy (14), as well as a blog post by Karpathy dedicated to underlying intuitions regarding RNNs (42).

Figure E2 shows output numbers for training and validation sets. Figures E3 and E4 show true-positive examples.

References

36. Paszke A, Gross S, Chintala S, et al. Automatic differentiation in PyTorch. https://openreview.net/pdf?id=BJJsrmfCZ. Published 2017. Accessed June 15, 2018.

37. Szegedy C, Vanhoucke V, Ioffe S, Shlens J, Wojna Z. Rethinking the inception architecture for computer vision. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016; 2818–2826.

38. Esteva A, Kuprel B, Novoa RA, et al. Corrigendum: dermatologist-level classification of skin cancer with deep neural networks. Nature 2017;546(7660):686 .

39. Haenssle HA, Fink C, Schneiderbauer R, et al. Man against machine: diagnostic performance of a deep learning convolutional neural network for dermoscopic melanoma recognition in comparison to 58 dermatologists. Ann Oncol 2018;29(8):1836–1842 .

40. Canziani A, Paszke A, Culurciello E. An analysis of deep neural network models for practical applications. arXiv [cs.CV] [preprint] https://arxiv.org/abs/1605.07678. Posted 2016. Accessed June 2018.

41. Russakovsky O, Deng J, Su H, et al. ImageNet Large Scale Visual Recognition Challenge. arXiv [cs.CV] [preprint] https://arxiv.org/abs/1409.0575. Posted 2014. Accessed June 2018.

42. Karpathy A. The unreasonable effectiveness of recurrent neural networks. http://karpathy.github.io/2015/05/21/rnn-effectiveness/. Published May 2015. Accessed June 2018.