Machine Learning in Nuclear Medicine: Part 2-Neural Networks and Clinical Aspects

Katherine Zukotynski[1*], Vincent Gaudet[2*], Carlos F. Uribe[3], Sulantha Mathotaarachchi[4], Kenneth C. Smith[5], Pedro Rosa-Neto[4], François Bénard[3,6], Sandra E. Black[7]

[*]*Contributed equally*

[1]Departments of Medicine and Radiology, McMaster University, Hamilton, ON, Canada

[2]Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, ON, Canada

[3]PET Functional Imaging, BC Cancer, Vancouver, BC, Canada

[4]Translational Neuroimaging Lab, McGill University, Montreal, QC, Canada

[5]Department of Electrical and Computer Engineering, University of Toronto, Toronto, ON, Canada

[6]Department of Radiology, University of British Columbia, Vancouver, BC, Canada

[7]Department of Medicine (Neurology), Sunnybrook Health Sciences Centre, University of Toronto, Toronto, ON, Canada

Corresponding author:
Katherine Zukotynski
Departments of Medicine and Radiology
McMaster University, Hamilton, ON
Telephone: 905-521-2100 x76556
Fax: 905-546-1125
zukotynk@mcmaster.ca

Word Count: 6071

Disclosure: There is no conflict of interest that will directly or indirectly influence the content of the manuscript submitted.

Running Title: Part 2: Neural networks

ABSTRACT

This article is the second part in our machine learning series. Part 1 provided a general overview of machine learning in nuclear medicine. Part 2 focuses on neural networks. We start with an example illustrating how neural networks work and a discussion of potential applications. Recognizing there is a spectrum of applications, we focus on recent publications in the areas of image reconstruction, low-dose PET, disease detection and models used for diagnosis and outcome prediction. Finally, since the way machine learning algorithms are reported in the literature is extremely variable, we conclude with a call to arms regarding the need for standardized reporting of design and outcome metrics and we propose a basic checklist our community might follow going forward.

Keywords: Machine Learning, Nuclear Medicine, Neural Networks

Learning Objectives

1. Provide an introduction to neural networks.
2. Discuss potential applications of neural networks with illustrative examples and figures.

Knowledge Acquisition

Upon review of this paper, the reader should be familiar with neural networks and should have an understanding of where they can be helpful in clinical practice.

INTRODUCTION

Part 1 in our series on machine learning (ML) in nuclear medicine (*1*) provided a general overview of ML algorithms and their basic components. While applications of ML algorithms such as random forests (*2-4*) and support vector machines (*5,6*) continue to proliferate, sophisticated ML algorithms such as artificial neural networks (ANNs) are becoming ubiquitous. Further, ANNs using radiomic data are increasingly common in nuclear medicine applications. Radiomic data typically refers to quantitative data from medical images, such as texture values enabling assessment of tumour heterogeneity, extracted either manually or using a computer-based approach (*7*). Part 2 provides an expanded explanation of ANNs, one of the most powerful ML models used today. After a brief review of ANN concepts introduced in Part 1, we illustrate how ANNs work using an example and follow this with a brief discussion of clinical applications.

BRIEF REVIEW FROM PART 1

ANNs are advanced ML algorithms (Fig. 1) typically used in classification (discrete-output) or regression (analog-output) applications. Although ANNs have existed for decades, they have only recently become common in medical imaging, in part due to technological advances as well as access to large datasets for training. Data input to an ANN is processed in steps, where each step consists of a layer of neurons. A neuron is a computational unit that: 1. Produces a weighted summation of input data, 2. Applies a bias, and 3. Computes a nonlinear transformation of the result. The output data from each layer passes to the next layer until the final layer produces the output result. The architectural design of an ANN describes the relationship between the various neurons and layers. ANNs are typically supervised, using tagged data to learn weights and biases, and can be simple including only a few layers and one output, or complex. More complex ANNs generally have greater capabilities but at higher computational cost. Complex ANNs are used for deep learning. Designing an ANN of optimal complexity to solve a specific task and obtaining access to sufficient high-quality input data, is challenging. Today, ANNs are

among the most common ML algorithms used in nuclear medicine and understanding how they work is key.

In this text we try to convey the structure and purpose of an ANN. To illustrate, the next section starts with an example of a simple ANN (with one layer) that could detect a handwritten letter from an input image. We then discuss more complex ANNs and their applications in nuclear medicine. For reference, common ML terms are summarized in Table 1. Further, when we write "nuclear medicine" please note we implicitly include PET, PET/CT and PET/MR.

ANNs: UNDERSTANDING BY EXAMPLE

In this example, we design and train an ANN with only one layer to recognize a handwritten image of the letter 'X' (Fig. 2). The input to the ANN is an 8x8 (2-dimensional) black-and-white image or 64-pixel array/vector (a_1-a_{64}), where each pixel has value: -1 (white) or 1 (black) (Fig. 2A and Fig. 2B). The output (f) is an analog number between 0 and 1 that reflects the likelihood the input is letter 'X'. For example, if the input is an image of 'X', the output should be 1; if the input is another letter it should be 0. Our ANN is trained using a dataset including several input images that have been tagged as representing 'X' or not (Fig. 2B). The ANN output is calculated by: 1. Multiplying each input by a corresponding weight (w_1-w_{64}), 2. Adding the products together (assuming all biases are 0), and 3. Passing the result through a nonlinear function (here a sigmoid function) called an *activation function* (Fig. 2C).

ANNs can have many types of activation functions, including a sigmoid function and a rectified linear unit (ReLU) (Fig. 2D). Each activation function constrains the output in some manner, eg. the sigmoid function constrains outputs to be between 0 and 1; the ReLU zeroes out negative numbers (Fig. 2E). These nonlinear functions are key for optimizing ANN performance.

Before the ANN can process new images, it must learn the values for the weights through training. To do this, the algorithm uses a cost (or loss) function that calculates how closely the model predicts the output for a particular training case. The ultimate goal of training is to minimize the cost function. A common cost function is to compute the error (E) between the trained (f_{train}) and desired

output (*f*), possibly the absolute difference between them (the square of the difference, and classification accuracy, are also common cost functions):

$$E = |f - f_{train}| \qquad (1).$$

The values of w_1-w_{64} that give the best performance are obtained by iterating through the training cases: The weights are initialized, a training case is input to the ANN, the error function is calculated, the weights are adjusted to nudge the ANN towards a lower cost, and a new training case is presented to the ANN. The process is done iteratively until the learned weights give a satisfactory cost.

Weight adjustment is often done using a gradient-descent algorithm, such as stochastic gradient descent. The gradient of the cost function is calculated, essentially the partial derivative (ie. slope) of the cost with respect to each weight. Once the partial derivatives are known, the weights are adjusted in the direction of steepest descent. However, there is no guidance as to how much weight adjustment is needed. Too little adjustment and little progress is made towards the end-goal; too much adjustment and the output might degrade. Consider a function *E(x)*, where we are trying to identify a minimal point (Fig. 3). If we start from point A, we should move right. However, a large step (point D) moves us too far.

For our ANN to detect the letter "X", we derive the partial derivative of *E* with respect to each weight. First, we express the output *f* based on the input pixels a_i and weights w_i:

$$f = \frac{1}{1+e^{-\sum_{i=1}^{64} a_i u w_i}} \qquad (2)$$

Taking the partial derivative of (1) with respect to w_i, gives:

$$\frac{\partial E}{\partial w_i} = \frac{(f_{train} - f)}{E} \frac{(e^{-\sum_{j=1}^{64} a_j w_j})}{(1+e^{-\sum_{j=1}^{64} a_j w_j})^2} a_i \qquad (3).$$

Each partial derivative shows the amount by which its corresponding weight should be adjusted per learning iteration.

We now train the ANN using 24 cases (handwritten samples): 12 of the letter 'X' and 12 of other letters. After training, the weights look like the 8x8 matrix shown in Fig. 4. The grayscale representation of weights in our ANN resembles an 'X' (Fig. 4A). This makes intuitive sense: since the weights reflect a probability map, an image of the weights resembles what the ANN is trained to detect.

Complex ANNs: Number of Layers and Architecture Design

ANNs capable of deep learning typically have many layers (*8*). Consider the processing involved with your brain reading this text as an example of this multi-step processing. It *might* go as follows: 1. Input an image through your eyes to your brain, 2. Your brain identifies strokes and puts strokes together determining how they form a pattern, 3. You recognize the pattern (or character), 4. You assemble neighboring characters and identify words, 5. Words come together into sentences, 6. Meaning is extracted from sentences, and 7. You process information and perform an analysis. While a programmer interested in deep learning might create a complex ANN, the task done at each layer is often not predefined by the programmer. Rather, the ANN operates for all intents and purposes as a "black-box". An ANN with more layers might be able to learn more but would also likely necessitate higher computational power possibly using graphics processing units (GPUs) or a remote server over the cloud. Some ANNs have over 100 layers, and millions of weights to optimize. The challenge is to build an ANN to solve a problem with a small number of operations, through optimizing architectural design.

Convolutional Neural Networks

Convolutional neural networks (CNNs) are a type of ANN where layers are structured in such a way that a convolutional kernel can be applied, which is important for image processing. A convolution is a common mathematical function, and a kernel refers to a matrix of weights that can be either pre-set, or

more commonly learned in the case of a CNN with access to training data. CNNs take a series of medical images, often single or multi-modality, as input, perform operations, calculate weights and biases for different layers and optimize parameters to minimize cost based on the desired output. Typically, input regions of interest (ROI) or features are not required. While simple ML algorithms can process images at the pixel level, CNNs have greater capacity for complex decision making and often outperform them, eg. in terms of classification accuracy.

Two common CNN architectural designs are illustrated in Fig. 5. For applications such as image reconstruction or segmentation, where the desired output is an image, variations on an encoder-decoder architecture are commonly used (Fig. 5A). An encoder reduces input data in a stepwise process to identify components or features. This can be accomplished through the use of the concept of stride, defined in Table 1 and illustrated in Fig. 6, or pooling defined in Table 1. A decoder then builds the output image from the features using a stepwise process, possibly including interpolation or up-sampling to increase resolution. Some architectures that follow this style include U-Net (2-dimensional data) and V-Net (3-dimensional data). For applications such as disease detection where the output is a classification (eg. disease present or absent), an ANN might only have an encoder phase where input data is reduced in a stepwise process that leads to the output classification (Fig. 5B). Res-Net is one such architecture.

Hardware Aspects

ANNs are typically programmed using software languages such as Matlab or Python. However, the hardware on which these programs run can significantly affect their speed. Simple ANNs can easily run on a standard laptop. However, more complex ANNs often need powerful hardware. GPUs have emerged as an effective hardware solution for ANNs since they are capable of performing many simple computations simultaneously, which improves speed. Sometimes, GPUs can be so powerful that they can perform all of the computations required for a convolution operation simultaneously. When ANNs get to be large enough that even a single GPU is insufficient, compute clusters (supercomputers often consisting of banks of GPUs) in data centers accessed over the cloud, may be needed.

NEURAL NETWORKS IN NUCLEAR MEDICINE: A SPECTRUM OF APPLICATIONS

Complex ANNs are used across a spectrum of nuclear medicine applications. A search of "machine learning" on PubMed returned 595 papers in 2009, 2402 in 2014, and 11297 in 2019, several including the use of ANNs. ANNs can help with image reconstruction or to create standard-dose from low dose images, as well as to improve scatter and attenuation correction (*9-19*). ANNs can also assist with disease detection and segmentation (*20-27*), disease diagnosis and outcome predictions (*28-33*). In this paper we have chosen to focus on a few applications with specific examples.

Neural Networks used for Image Reconstruction and Low-Dose PET

Signal noise is inherent in nuclear imaging and may be aggravated by using low-dose techniques or reducing image acquisition time. CNNs can be used during image reconstruction to generate higher quality images compared to conventional techniques and to improve the perceived quality of noisy images. An array of architectural designs may be used (and details in the literature are often limited).

One approach, focussing on image reconstruction is illustrated by Häggström, et al. (*11*). The authors programmed a CNN using an encoder-decoder architecture, similar to that presented in Fig. 5A, to reconstruct PET images from data synthesized almost entirely using a combination of phantom, simulation and augmentation techniques. The input to the CNN was PET sinogram data represented by a 288x289x1 matrix; the output was image data represented by a 128x128x1 matrix. The encoder reduced the input data through sequential layers applying convolution kernels with decreasing kernel size and stride 2, and activation functions including batch normalization and ReLUs. The decoder up-sampled the data using sequential layers to apply convolution kernels, increase matrix size, and apply activation functions including batch normalization and ReLUs to produce the final PET images. Several design modifications were studied, including differing numbers of layers and kernel size, among others. The CNN was able to generate PET images with higher quality compared to techniques such as ordered subset expectation maximization (OSEM) or filtered back projection (FBP).

Often the CNN includes several layers with parallel paths (also referred to as parallel channels) to apply a host of specific kernels to dissect out certain image features, and then combine feature information through a series of layers to generate the noiseless output image. Sometimes the input is a noisy image and the CNN is designed to reduce this input data to a series of low-resolution images that identify abstract features such as edges, texture, etc., and then progressively reconstruct a noiseless output image at the same resolution as the input image. These CNNs typically undergo supervised training using pairs of noisy input and noiseless output images. The key is to ascertain no significant information is lost or false information added.

As an illustration Chen, et al. (*12*) used a CNN with a U-Net architecture, similar to that presented in Fig. 5A, to synthesize full-dose ^{18}F-Florbetaben PET/MR images from low-dose images obtained using 1% of the raw list-mode PET data. The quality of the synthesized images was subjectively evaluated on a 5-point scale by 2 readers, while Bland-Altman plots were used to compare standardized uptake value ratios (SUVRs). The authors found the synthesized images showed improved quality metrics compared with low-dose images, with high accuracy for amyloid status and similar intrareader reproducibility compared with full-dose images. A review of CNN approaches for handling low-dose PET is given in (*34*).

Neural Networks used for Disease Detection and Segmentation

A common application of neural networks is disease detection and segmentation, for example to quantify disease burden. A time-consuming task in practice, essentially, this is a pixel-wise classification problem: each pixel must be tagged as normal/abnormal and joined to the region it belongs (e.g., liver, spleen…). Typically, the output is an image(s) at the same resolution as the input image(s), with feature information extracted by the neural network used to create overlying segmentation images. Similar to denoising, input and output images are at the same resolution and training is usually supervised, using combinations of raw and segmented images. Several papers have been written on lesion detection and segmentation using neural networks (*20-26*) with differing architectural designs, although often a U-Net.

As illustration, consider the paper by Zhao et al. (*27*). The authors created CNNs with the aim of automatically segmenting sites of disease on ^{68}Ga-PSMA-11 PET/CT images, to provide a yes-no answer as to whether a voxel reflected a lesion. The overall framework consisted of 2 components operating in series: 1) 3 parallel CNN paths each designed to detect lesions in one of 3 different planes, and 2) per-voxel final majority voting based on intermediate decisions from each plane's CNN. The CNNs had a U-Net structure consisting of an "encoding stack" followed by a "decoding stack" that fused feature maps with original images, similar in structure to Fig. 5A. The encoding stack included 3x3 convolutions, 2x2 max pooling with stride 2 for down sampling, ReLU and batch normalization. The decoding stack synthesized the information using a transposed convolution with kernel size 2x2 and stride 2, a concatenation operation and 3x3 convolutions with ReLU and batch normalization. At the last layer of the CNN, the sigmoid function helped map features to a segmentation probability map. The Dice similarity coefficient (DSC) was used to evaluate the accuracy of anatomic segmentation. ^{68}Ga-PSMA-11 PET/CT scans from 193 men with metastatic castration-resistant prostate cancer (mCRPC) were randomly divided into 130 training scans and 63 testing scans. All lesions in the pelvis were manually delineated (ie. 1003 bone lesions and 626 lymph node lesions, among others). A fivefold cross-validation was used for optimization. Using the manually annotated images as ground truth, a lesion was considered to be correctly detected when the overlap ratio exceeded a threshold of 10%. The detection accuracy, sensitivity and F1-score (harmonic mean of accuracy and sensitivity) were 0.99, 0.99, 0.99 for bone lesions respectively, and 0.94, 0.90, 0.92 for lymph nodes respectively. The image segmentation accuracy was lower than the lesion detection accuracy. The overall model achieved average DSCs (65%, 54%), PPV (80%, 67%) and specificity (61%, 55%) for bone and lymph node lesions respectively.

While the possibility of using ANNs for lesion detection and image segmentation has enormous impact for clinical practice, manual assessment is still often used.

Neural Networks used for Disease Diagnosis and Outcome Prediction

ANNs can assist with disease diagnosis and outcome prediction (*28-32*). Often, only a small set of input images/data are needed and models that input full-resolution images or several data sources gradually reduce this to distill a diagnosis or outcome by the final layer. Typically, these are classification problems, training is supervised and often a Res-Net architectural design is used, similar to that presented in Fig. 5B. While early results are promising, rigorous evidence supporting ML models is lacking. A systematic review by Nagendran at al. published this year found 1 randomized clinical trial related to breast ultrasound and 2 non-randomized prospective studies investigating intracranial hemorrhage (*35*). The field is young and it is important to remember to temper our claims of imminent clinical impact.

Mayerhoefer et al. provides an illustration of a neural network use for a predictive application (*33*). Specifically, the authors proposed to determine if radiomic features on ^{18}F-FDG PET/CT alone or in combination with clinical, laboratory and biological parameters were predictive of 2-year progression-free survival (PFS) in subjects with mantle cell lymphoma (MCL). A multilayer feed-forward neural network was used, which relied on a back-propagation learning algorithm (*8*) in combination with logistic regression analysis for feature selection. Few specific details are given, although we are told there was a minimum of 1 hidden layer with a minimum of 3 neurons per hidden layer. The input included a guess of weights for individual radiomic features and the classification step was repeated 5 times. The data consisted of 107 ^{18}F-FDG PET/CT scans in treatment-naïve MCL patients with baseline and follow-up data to the date of progression, death or a minimum of 2 years. Cases were randomly split into 75 training and 32 validation cases for each classification step repetition. A semiautomatic process was used for lesion delineation and several parameters were included for analysis: SUVmax, SUVmean, SUVpeak, total lesion glycolysis (TLG) and 16 textural features derived from the grey-level co-occurrence matrix calculated in 3D. Outcome measures included the area under the receiver operating characteristic (ROC) curve (AUC) and classification accuracy. While radiomic features were not significantly correlated with absolute PFS (in months), 2-year PFS status correlated with SUVmean ($p=0.022$) and Entropy ($p=0.034$) in a multi-variate analysis. When SUVmean and Entropy values were input to the neural network, AUCs for 2-year PFS prediction were 0.70-0.73 (median 0.72), with classification accuracies 71.0-76.7%

(median 74.4%) in training and 70.6-86.8% (median 74.3%) in validation cases, improving when combined with additional clinical/laboratory/biological data.

Common Themes and a Call to Arms

Ultimately, we arrive at a few conclusions regarding ANNs in nuclear medicine:

1. Good performance is often achieved with less than 10 layers. Many papers use data from small patient cohorts (~20-200) supplemented with data augmentation techniques to generate larger training and/or cross-validation datasets or generate data using simulation software.
2. The computational cost of a ML algorithm is rarely reported yet should not be ignored as it directly impacts reproducibility and clinical practicality. Those papers that do describe the algorithm structure, often omit key information making it nearly impossible for a reader to recreate the model. FLOPs, the cost metric commonly used by computer scientists and engineers, is rarely reported.
3. There is a lack of well conducted, systematic studies, with few to no randomized clinical trials evaluating applications in routine clinical practice.

We are in the early days of the application of ML to nuclear medicine, and it is becoming evident there is a need for the community to come together and design standard elements of reporting needed for the field's evolution. This would make it easier to assess algorithm effectiveness, cost and appropriate use. If we had the details, we could graph metrics of input, algorithm complexity and output to establish algorithms that are most effective for a specific task. As a starting point for discussion, Fig. 7 portrays a conceptual graph that could be plotted if standardized details of algorithms were reported, and which would provide insight into trends. The graph uses the example of low-dose PET, and plots percentage dose versus ML algorithm computational cost. Any paper that reports dose, computational cost, and algorithm family, could be included as a point on the graph. As more data becomes available, we would see trend lines emerge, such as the dashed lines shown representing constant classification accuracy for

algorithm family. Bounds on algorithm family capability might be inferred. For example, Minark et al. (*10*) report performance results of a CNN at various image noise levels (analogous to percentage dose). While the CNN performs well, the computational cost is not reported, and it difficult to exactly replicate what was done. With additional information, we could have plotted several points on our graph, upon which to base future work.

To gather insight into algorithms best suited for a given task, and the computational cost needed to achieve a desired output, we advocate our community use a checklist for reporting ML algorithms. Table 2 provides our top 5 points to include. We hope this represents a start for further discussion.

CONCLUSIONS

We are witnessing a potentially phenomenal development in clinical nuclear medicine. While ANNs are becoming ever more common in nuclear medicine, new families of algorithms are being developed. Further, as databases of shared images continue to be created, there will be expanding datasets useful for training, validation and testing purposes. Several issues remain, notably those surrounding ethics and privacy of data collection, de-identification and ownership. In some situations, it may prove easier to download an algorithm to multiple sites instead of uploading multi-site data to a communal database. Regardless, to understand where we are, a standardized practice for reporting ML algorithm metrics would be helpful. We present a list of our top 5 items to include (Table 2) and suggest how data could be compiled to generate graphs showing which family of ML algorithms might be best suited for a given application. We hope this paper has provided insight into how ANNs work, the spectrum of clinical tasks they can help with, and where we might go from here.

REFERENCES

[1] Uribe CF, Mathotaarachchi S, Gaudet V, et al. Machine learning in nuclear medicine: Part 1-Introduction. *J Nucl Med*. 2019;60:451-458.

[2] Zukotynski K, Gaudet V, Kuo PH, et al. The use of random forests to classify amyloid brain PET. *Clin Nucl Med*. 2019;44:784-788.

[3] Nuvoli S, Spanu A, Fravolini ML, et al. [123]Metaiodobenzylguanidine (MIBG) cardiac scintigraphy and automated classification techniques in Parkinsonian disorders. *Mol Imaging Biol*. 2020;22:703-710.

[4] Perk T, Bradshaw T, Chen S, et al. Automated classification of benign and malignant lesions in ^{18}F-NaF PET/CT images using machine learning. *Phys Med Biol*. 2018;63:225019.

[5] Nicastro N, Wegrzyk J, Preti MG, et al. Classification of degenerative parkinsonism subtypes by support-vector-machine analysis and striatal ^{123}I-FP-CIT indices. *J Neurol*. 2019;266:1771-1781.

[6] Kim JP, Kim J, Kim Y, et al. Staging and quantification of florbetaben PET images using machine learning: impact of predicted regional cortical tracer uptake and amyloid stage on clinical outcomes. *Eur J Nucl Med Mol Imaging*. 2020;47:1971-1983.

[7] Mayerhoefer ME, Materka A, Langs G, et al. Introduction to radiomics. *J Nucl Med*. 2020;61:488-495.

[8] LeCun Y, Bengio Y, Hinton G. Deep learning. *Nature*. 2015;521:436-444.

[9] Xiang L, Qiao Y, Nie D, An L, Wang Q, Shen D. Deep auto-context convolutional neural networks for standard-dose PET image estimation from low-dose PET/MRI. *Neurocomputing*. 2017;267:406-416.

[10] Minarik D, Enqvist O, Trägårdh E. Denoising of scintillation camera images using a deep convolutional neural network: A Monte Carlo simulation approach. *J Nucl Med*. 2020;61:298-303.

[11] Häggström I, Schmidtlein CR, Campanella G, Fuch TJ. DeepPET: A deep encoder-decoder network for directly solving the PET image reconstruction inverse problem. *Med Image Anal*. 2019;54:253-262.

[12] Chen KT, Gong E, de Carvalho Macruz FB, et al. Ultra-low-dose (18)F-florbetaben amyloid PET imaging using deep learning with multi-contrast MRI inputs. *Radiology*. 2019;290:649–656.

[13] Gao F, Shah V, Sibille L, Zuehlsdorff S. An AI system to determine reconstruction parameters and improve PET image quality. *J Nucl Med*. 2018;59:31.

[14] Hwang D, Kim KY, Kang SK, et al. Improving the accuracy of simultaneously reconstructed activity and attenuation maps using deep learning. *J Nucl Med*. 2018;59:1624-1629.

[15] Leynes AP, Yang J, Wiesinger F, et al. Zero-echo-time and dixon deep pseudo-CT (ZeDD CT): direct generation of pseudo-CT images for pelvic PET/MRI attenuation correction using deep convolutional neural networks with multiparametric MRI. *J Nucl Med*. 2018;59:852–858.

[16] Hwang D, Kim KY, Kang SK, et al. Improving the accuracy of simultaneously reconstructed activity and attenuation maps using deep learning. *J Nucl Med*. 2018;59:1624–1629.

[17] Spuhler KD, Gardus J 3rd, Gao Y, DeLorenzo C, Parsey R, Huang C. Synthesis of patient-specific transmission data for PET attenuation correction for PET/MR neuroimaging using a convolutional neural network. *J Nucl Med*. 2019;60:555-560.

[18] Torrado-Carvajal A, Vera-Olmos J, Izquierdo-Garcia D, et al. Dixon-VIBE deep learning (DIVIDE) pseudo-CT synthesis for pelvis PET/MR attenuation correction. *J Nucl Med*. 2019;60:429-435.

[19] Gong K, Guan J, Kim K, et al. Iterative PET image reconstruction using convolutional neural network representation. *IEEE Trans Med Imaging*. 2019;38:675-685.

[20] Belal SL, Sadik M, Kaboteh R, et al. Deep learning for segmentation of 49 selected bones in CT scans: First step in automated PET/CT-based 3D quantification of skeletal masses. *Eur J Radiol*. 2019;113:89-95.

[21] Gsaxner C, Roth PM, Wallner J, Egger J. Exploit fully automatic low-level segmented PET data for training high-level deep learning algorithms for the corresponding CT data. *PLoS ONE*. 2019;14:e0212550.

[22] Huang B, Chen Z, Wu PM, et al. Fully automated delineation of gross tumor volume for head and neck cancer on PET-CT using deep learning: a dual-center study. *Contrast Media Mol Imaging*. 2018;2018:8923028.

[23] Zhao X, Li L, Lu W, Tan S. Tumor co-segmentation in PET/CT using multi-modality fully convolutional neural network. *Phys Med Biol*. 2019;64:015011.

[24] Hatt M, Laurent B, Ouahabi A, et al. The first MICCAI challenge on PET tumor segmentation. *Med Image Anal*. 2018;44:177–95.

[25] Bi L, Kim J, Kumar A, Wen L, Feng D, Fulham M. Automatic detection and classification of regions of FDG uptake in whole-body PET-CT lymphoma studies. *Comput Med Imaging Graph*. 2017;60:3–10.

[26] Xu L, Tetteh G, Lipkova J, et al. Automated whole-body bone lesion detection for multiple myeloma on ^{68}Ga-pentixafor PET/CT imaging using deep learning methods. *Contrast Media Mol Imaging*. 2018;2018:11.

[27] Zhao Y, Gafita A, Vollnberg B, et al. Deep neural network for automatic characterization of lesions on 68Ga-PSMA-11 PET/CT. *Eur J Nucl Med Mol Imaging*. 2020;47:603-613.

[28] Commandeur F, Goeller M, Razpour A, et al. Fully automated CT quantification of epicardial adipose tissue by deep learning: A multicenter study. *Radiol Artific Intell*. 2019;1:e190045

[29] Eisenberg E, Commandeur F, Chen X, et al. Deep learning-based quantification of epicardial adipose tissue volume and attenuation predicts major adverse cardiovascular events in asymptomatic subjects. *Circ Cardiovasc Imaging*. 2020;13:e009829

[30] Hartenstein A, Lubbe F, Baur ADJ, et al. Prostate cancer nodal staging: using deep learning to predict 68Ga-PSMA-Positivity from CT imaging alone. *Sc Rep*. 2020;10:3398.

[31] van Velzen SGM, Lessmann N, Velthuis BK, et al. Deep learning for automatic calcium scoring in CT: Validation using multiple cardiac CT and chest CT protocols. *Radiology*. 2020;295:66-79.

[32] Huang Y, Xu J, Zhou Y, et al. Diagnosis of Alzheimer's Disease via multi-modality 3D convolutional neural network. *Front Neurosci*. 2019;13:509-520

[33] Mayerhoefer ME, Riedl CC, Kumar A, et al. Radiomic features of glucose metabolism enable prediction of outcome in mantle cell lymphoma. *Eur J Nucl Med Mol Imaging*. 2019;46: 2760-2769.

[34] Zaharchuk G. Next generation research applications for hybrid PET/MR and PET/CT imaging using deep learning. *Eur J Nucl Med Mol Imag*. 2019;46:2700-2707.

[35] Nagendran M, Chen Y, Lovejoy CA, et al. Artificial intelligence versus clinicians: a systematic review of design, reporting standards, and claims of deep learning studies. *BMJ*. 2020;368-379.

Figure 1. Venn diagram depicting the relationship of ML, AI and deep learning. Simple ML algorithms such as random forests and K-means clustering are shown. Complex ML algorithms such as ANNs extend beyond supervised deep learning (where they are primarily used). Algorithms that are neither supervised nor unsupervised, e.g., reinforcement learning, are not shown.

Figure 2. A. Input: 8x8 matrix (a_1-a_{64}) of pixels where each pixel is -1 (white) or +1 (black). B. Examples from a training dataset of handwritten letters mapped to a binary input. C. Single-layer neural network with input shown in (A), 64-weights (w_1-w_{64}), sigmoid activation function, and one output (*f*) that is an analog number between 0 and 1 reflecting the probability the input is 'X'. D. Mathematical expressions for activation functions. E. Graphs showing sigmoid (blue) and ReLU (red); for the sigmoid the output is constrained to be between 0 and 1; for the ReLU, negative inputs are zeroed.

Figure 3. Starting from point (A), we wish to find the lowest point in function *E(x)*, labeled (B). Suppose we know the slope of *E(x)* at point (A), a gradient-based search suggests we move right to a lower point. Ideally, we prefer a small step, to (C) rather than a large step, to (D).

Figure 4. A. Greyscale image of the matrix of weights (kernel) following 24 training cases. Darker colouring represents pixels with higher weights. Notice the image resembles an 'X'. B. New examples of images of handwritten letters input to the trained ANN show the letter 'X' is identified with 92% likelihood (ANN output f=0.92) and 'O' is not interpreted as 'X', with likelihood of 0% (f=0.00).

Figure 5. Illustration of ANN architectures. A. An encoder-decoder design is helpful for image segmentation. In the encoder, the input image resolution is reduced while the number of images increases. The first layer produces two images, the first by applying a 2x2 kernel using a convolutional operation with a stride of 2, and the second by applying a different kernel. Since one image is input, we denote these as 1x2x2 kernels. The second layer produces two output images, from two input images that are

treated as a volume, again using two different kernels, denoted as 2x2x2, with stride of 1. The third layer applies four different 2x2x2 kernels with stride of 2, to generate four images that are input to the fourth layer. In the decoder, up-sampling creates higher resolution images so the CNN input and output resolutions are similar. A feedforward path adds data from earlier layers. The U shape gives rise to the name U-Net. B. An encoder design is helpful for disease detection. Over consecutive layers, image resolution is decreased, to identify features that are encoded into feature maps. The final layer is often fully connected; the two outputs shown each use a weighted sum of every pixel from the preceding layer. Res-Nets are an example of this.

Figure 6. Illustration of stride. A. Input 8x8 matrix is processed in a convolutional layer with a 3x3 kernel (weights w_1-w_9). Each pixel in the output 8x8 matrix is calculated by multiplying the 9 nearest neighbors to the corresponding input pixel by respective kernel weights. As illustration, the calculation for output pixel f_{45} is shown. B. Using a stride of 2, every second output pixel is calculated in both dimensions, resulting in a 4x4 output image.

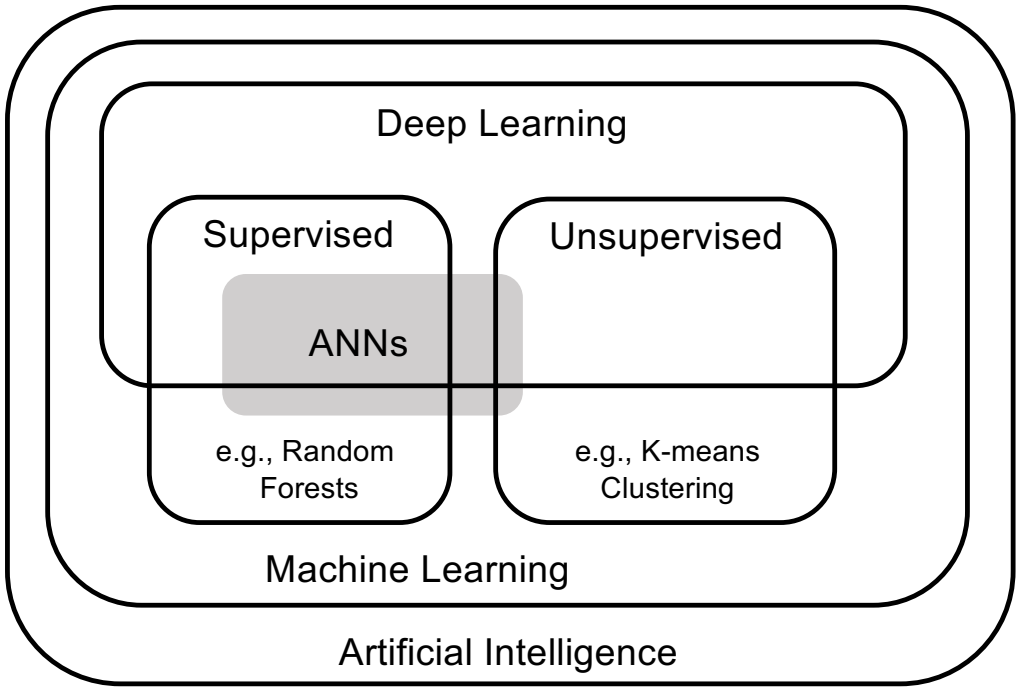
Figure 7. Conceptual graph showing how classification accuracy (dotted curves) and counts might be impacted by ML algorithm computational cost (and ability to learn complex tasks). Such graphs require researchers provide specific details about their ML implementations.

Table 1. Common Terms Encountered Discussing Neural Networks

Term	Explanation	Comment
Fully connected layer	Each input to a layer is used to compute each output from the layer.	Fig. 1C illustrates a fully connected layer with 64 inputs and 1 output. While the number of output data points could be smaller than the number of input data points, this is not required.
Kernel	Matrix of numbers in a CNN where the numbers are typically learned through exposure to a training dataset.	3x3 kernels or 3x3x3 kernels are common.
Stride	A number that represents how many pixels a kernel skips each time it processes an image in a CNN.	Fig. 5 illustrates stride. The output image has fewer pixels than the input image resulting in an output image represented by a matrix of lower dimension.
Pooling	Operation in a CNN that reduces image resolution by averaging or taking a maximum of a local region.	A pooling layer could have as input an image represented by a 128x128 matrix and produce as output an image represented by a 64x64 matrix. This could be accomplished by dividing the input matrix into 2x2 blocks and then reducing each block of 4 numbers to one number representing the maximum value.
FLOP	FLOP stands for FLOating-Point operation and represents a measure of computing power.	The FLOPs associated with a network typically refer to the computing power needed for the network to run after it has been trained. Using Fig. 1 as an illustration, there are 64 multiplications and 63 pairwise additions, representing 127 FLOPs (omitting the sigmoid function). A CNN might require billions of FLOPs, while a simple ML algorithm such as a random forest or support vector machine might require thousands.

Table 2. Suggested checklist to include for ML-related algorithm reporting

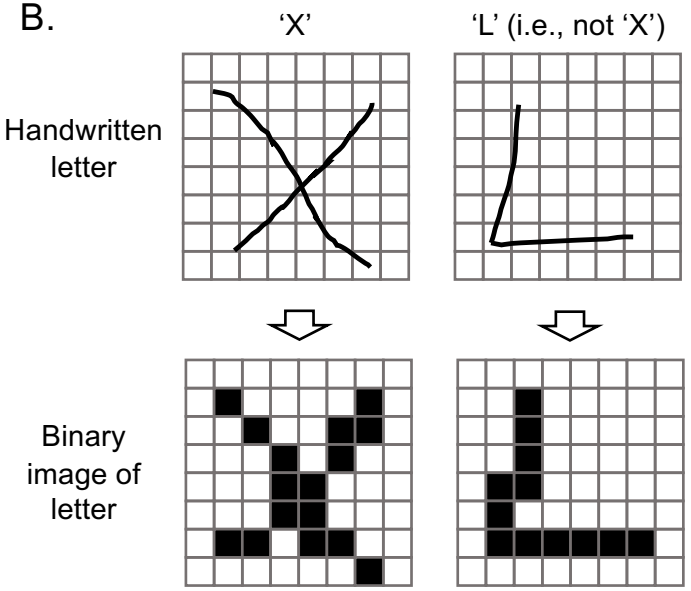
Question	**Possible Metric**	**Comment**
1. ML algorithm?	Family of ML algorithms	ie., CNN, random forest, support vector machine…
2. Architecture details?	Dependent on algorithm	ie., for a CNN report number of layers, kernel size, strides, and show a complete block diagram with sufficient detail that the model could be independently reconstructed.
3. Computational cost?	Number of parameters, FLOPs	ie., while consulting a computing expert, similar to consulting a statistician for clinical trials, is suggested, authors may generate this themselves.
4. Data?	Training, validation, testing	ie., data type, number of validation/testing cases, use of cross-validation, data source (algorithms trained with data from a single institution might not perform well using data from another institution).
5. Figure of merit?	Classification accuracy, dose reduction…	ie., key numerical performance results should be given such as classification accuracy... ultimately this should be standardized for a given application.


Deep Learning
Supervised
Unsupervised
ANNs
e.g., Random Forests
e.g., K-means Clustering
Machine Learning
Artificial Intelligence

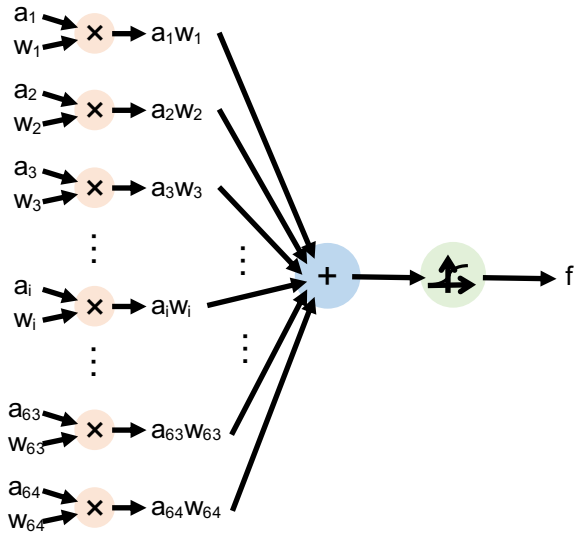
A.

a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8
a_9	a_{10}	a_{11}	a_{12}	a_{13}	a_{14}	a_{15}	a_{16}
a_{17}	a_{18}	a_{19}	a_{20}	a_{21}	a_{22}	a_{23}	a_{24}
a_{25}	a_{26}	a_{27}	a_{28}	a_{29}	a_{30}	a_{31}	a_{32}
a_{33}	a_{34}	a_{35}	a_{36}	a_{37}	a_{38}	a_{39}	a_{40}
a_{41}	a_{42}	a_{43}	a_{44}	a_{45}	a_{46}	a_{47}	a_{48}
a_{49}	a_{50}	a_{51}	a_{52}	a_{53}	a_{54}	a_{55}	a_{56}
a_{57}	a_{58}	a_{59}	a_{60}	a_{61}	a_{62}	a_{63}	a_{64}

B.



C.

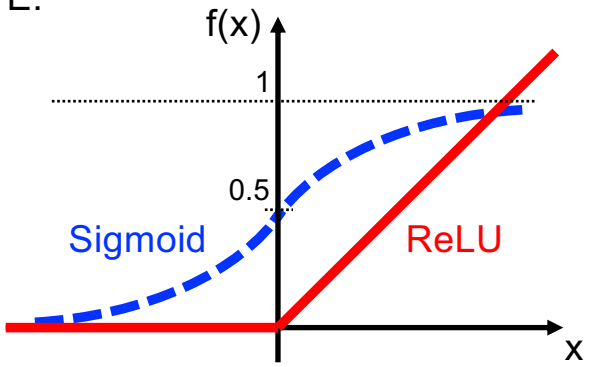


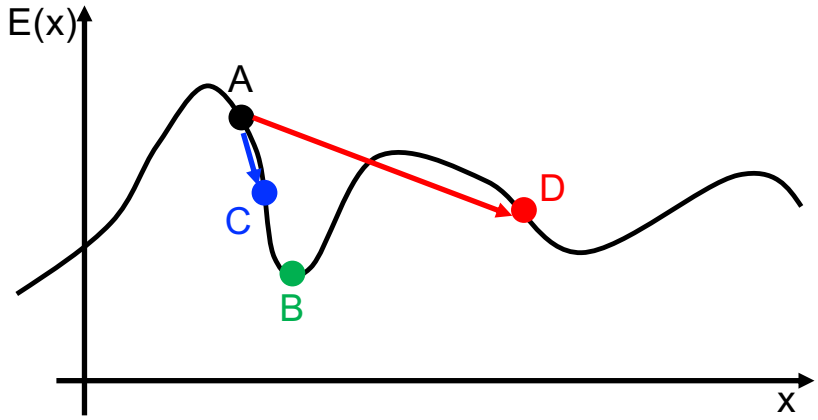
D.

Sigmoid: $f(x) = \dfrac{1}{1 + e^{-x}}$

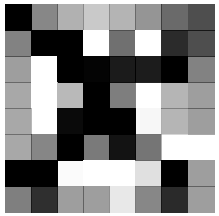
ReLU: $f(x) = \begin{cases} x, x > 0 \\ 0, x \leq 0 \end{cases}$

E.

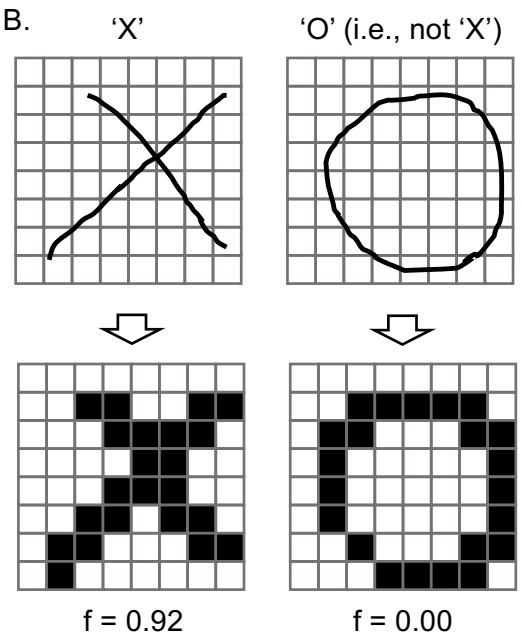


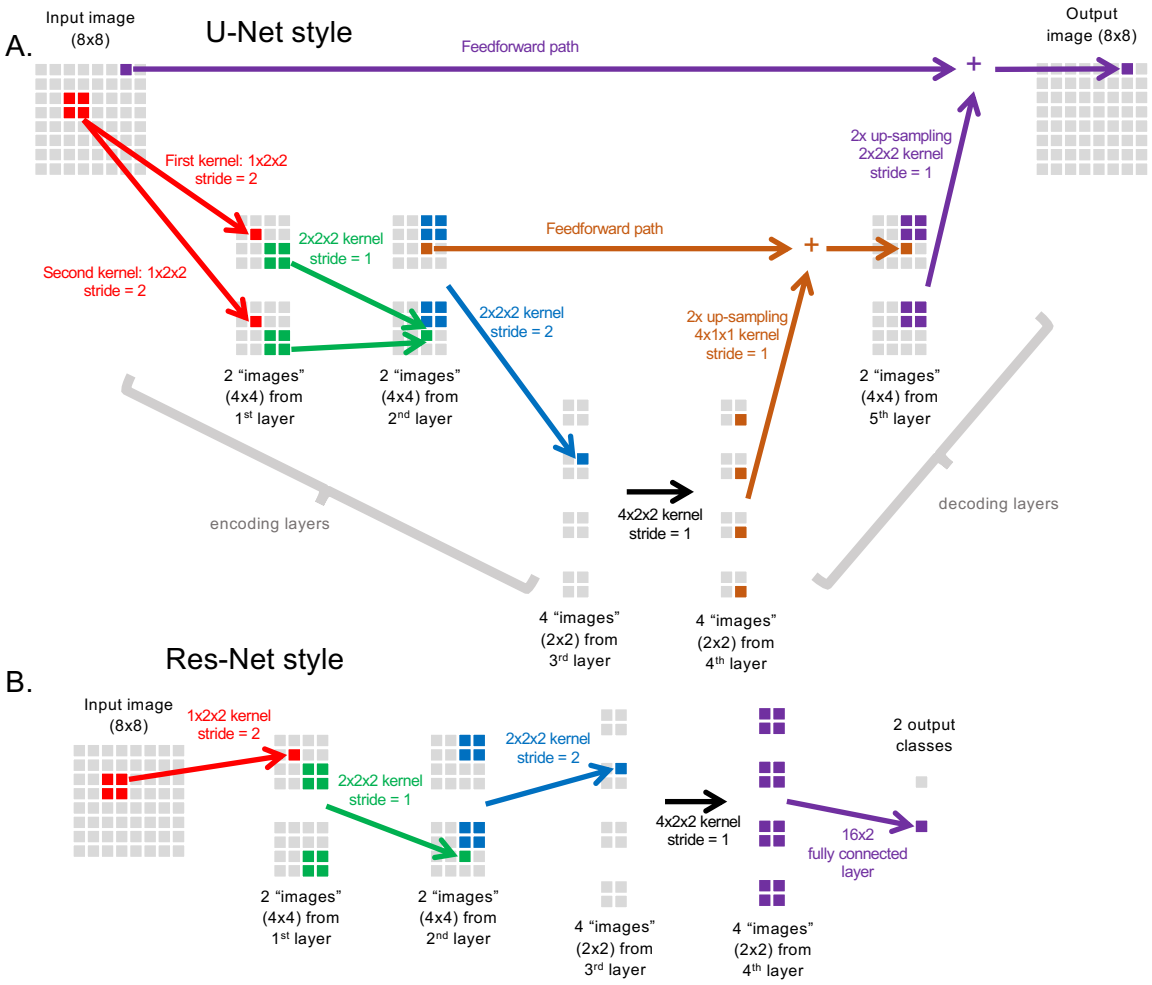

E(x)
A
C
B
D
x

A.



B.




A.
U-Net style
Input image
(8x8)
Feedforward path
Output
image (8x8)
+
First kernel: 1x2x2
stride = 2
2x2x2 kernel
stride = 1
Feedforward path
+
2x up-sampling
2x2x2 kernel
stride = 1
Second kernel: 1x2x2
stride = 2
2x2x2 kernel
stride = 2
2x up-sampling
4x1x1 kernel
stride = 1
2 “images”
(4x4) from
1st layer
2 “images”
(4x4) from
2nd layer
2 “images”
(4x4) from
5th layer
encoding layers
4x2x2 kernel
stride = 1
decoding layers
4 “images”
(2x2) from
3rd layer
4 “images”
(2x2) from
4th layer
B.
Res-Net style
Input image
(8x8)
1x2x2 kernel
stride = 2
2x2x2 kernel
stride = 1
2x2x2 kernel
stride = 2
2 output
classes
4x2x2 kernel
stride = 1
16x2
fully connected
layer
2 “images”
(4x4) from
1st layer
2 “images”
(4x4) from
2nd layer
4 “images”
(2x2) from
3rd layer
4 “images”
(2x2) from
4th layer

A.

Input image

a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8
a_9	a_{10}	a_{11}	a_{12}	a_{13}	a_{14}	a_{15}	a_{16}
a_{17}	a_{18}	a_{19}	a_{20}	a_{21}	a_{22}	a_{23}	a_{24}
a_{25}	a_{26}	a_{27}	a_{28}	a_{29}	a_{30}	a_{31}	a_{32}
a_{33}	a_{34}	a_{35}	a_{36}	a_{37}	a_{38}	a_{39}	a_{40}
a_{41}	a_{42}	a_{43}	a_{44}	a_{45}	a_{46}	a_{47}	a_{48}
a_{49}	a_{50}	a_{51}	a_{52}	a_{53}	a_{54}	a_{55}	a_{56}
a_{57}	a_{58}	a_{59}	a_{60}	a_{61}	a_{62}	a_{63}	a_{64}

Kernel (3x3)

w_1	w_2	w_3
w_4	w_5	w_6
w_7	w_8	w_9

Output image (8x8, stride = 1)

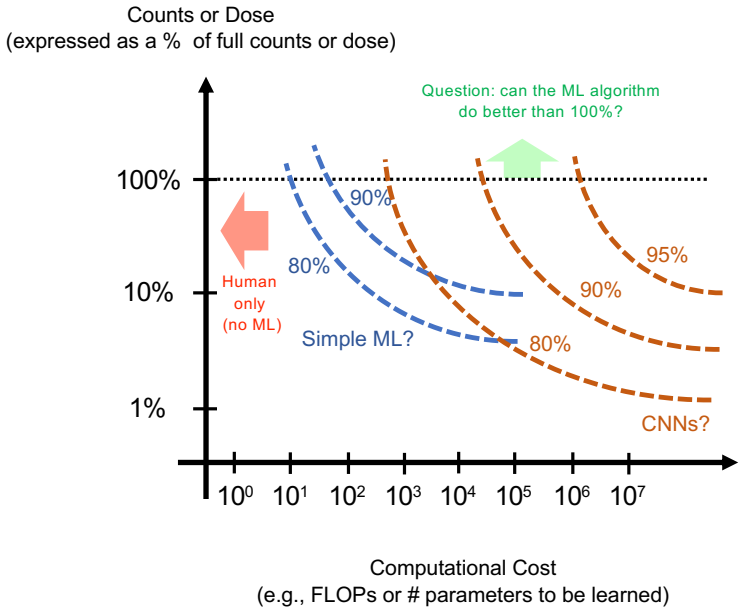
f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8
f_9	f_{10}	f_{11}	f_{12}	f_{13}	f_{14}	f_{15}	f_{16}
f_{17}	f_{18}	f_{19}	f_{20}	f_{21}	f_{22}	f_{23}	f_{24}
f_{25}	f_{26}	f_{27}	f_{28}	f_{29}	f_{30}	f_{31}	f_{32}
f_{33}	f_{34}	f_{35}	f_{36}	f_{37}	f_{38}	f_{39}	f_{40}
f_{41}	f_{42}	f_{43}	f_{44}	f_{45}	f_{46}	f_{47}	f_{48}
f_{49}	f_{50}	f_{51}	f_{52}	f_{53}	f_{54}	f_{55}	f_{56}
f_{57}	f_{58}	f_{59}	f_{60}	f_{61}	f_{62}	f_{63}	f_{64}

$$f_{45} = a_{36}w_1 + a_{37}w_2 + a_{38}w_3 + a_{44}w_4 + a_{45}w_5 + a_{46}w_6 + a_{52}w_7 + a_{53}w_8 + a_{54}w_9$$

B.

Output image (4x4, stride = 2)

f_1		f_3		f_5		f_7	
f_{17}		f_{19}		f_{21}		f_{23}	
f_{33}		f_{35}		f_{37}		f_{39}	
f_{49}		f_{51}		f_{53}		f_{55}	


Counts or Dose
(expressed as a % of full counts or dose)
Question: can the ML algorithm
do better than 100%?
100%
10%
1%
Human
only
(no ML)
90%
80%
Simple ML?
95%
90%
80%
CNNs?
10^0 10^1 10^2 10^3 10^4 10^5 10^6 10^7
Computational Cost
(e.g., FLOPs or # parameters to be learned)