Deep Learning for PET Image Reconstruction

Andrew J. Reader, Guillaume Corda, Abolfazl Mehranian, Casper da Costa-Luis, *Student Member, IEEE*, Sam Ellis, and Julia A. Schnabel, *Senior Member, IEEE*

***Abstract*—This article reviews the use of a subdiscipline of artificial intelligence (AI), *deep learning*, for the reconstruction of images in positron emission tomography (PET). Deep learning can be used either directly or as a component of conventional reconstruction, in order to reconstruct images from noisy PET data. The review starts with an overview of conventional PET image reconstruction and then covers the principles of general linear and convolution-based mappings from data to images, and proceeds to consider nonlinearities, as used in convolutional neural networks (CNNs). The direct deep-learning methodology is then reviewed in the context of PET reconstruction. Direct methods learn the imaging physics and statistics from scratch, not relying on *a priori* knowledge of these models of the data. In contrast, model-based or physics-informed deep-learning uses existing advances in PET image reconstruction, replacing conventional components with deep-learning data-driven alternatives, such as for the regularization. These methods use trusted models of the imaging physics and noise distribution, while relying on training data examples to learn deep mappings for regularization and resolution recovery. After reviewing the main examples of these approaches in the literature, the review finishes with a brief look ahead to future directions.**

***Index Terms*—Artificial intelligence (AI), deep learning, image reconstruction, machine learning, positron emission tomography (PET).**

I. Introduction

ARTIFICIAL intelligence (AI) is now having a widespread impact on many and diverse fields, including inverse problems [1]. AI is wide-ranging, and generally concerns algorithms for learning tasks of varying complexity (from autonomous driving through to filtering out spam emails). A specific subdiscipline of AI is referred to as *deep learning*, [2] which usually involves artificial neural network (ANN) mappings of inputs to outputs. Example inputs could be raw data from sensing devices, and example outputs could be classifications, processed results or images enhanced for particular tasks. The reasons for referring to these mappings as deep learning, as part of AI, are that 1) the mappings usually involve a cascaded series of operators (with their own inputs and outputs) known as layers, giving the notion of *depth* and 2) the operators use parameters which are *learned* from example training datasets. In the training datasets, for the case of supervised learning, example inputs are paired with their corresponding desired outputs. For unsupervised learning, the training data may consist of example inputs only (for learning of latent representations of the data [3]), or of unpaired example inputs and example outputs [4]. A further category, that of self-supervised learning [5], [6], needs only input data examples and instructions on how to create labels (rather than providing labels) thus reducing the need for human interaction with the learning process. In the context of paired inputs and outputs (whether supervised one to one pairings, or an unsupervised pair of distributions of data), the mapping learned between the domains can then be subsequently used on entirely new, never before seen input data, in order to predict the output. Conversely, in the context of unsupervised learning for a single dataset, the learned mapping can be used to generate or reconstruct images which are restricted to lie within a limited subspace/manifold/domain, corresponding to the same subspace from which the training data were sampled [7].

While ANNs have been applied to reconstruction in emission tomography from as early as 1991 [8], it was only with various technical advances in optimization capabilities (made available in deep learning toolboxes, such as TensorFlow, originating from Google, and PyTorch, originating from FaceBook) and the demonstrated success of deep learning in other fields (such as object recognition from ImageNet data in 2009 [9]) that eventually, from ∼2017, deep learning reached the world of medical image processing [10] and reconstruction in emission tomography. The earliest examples for medical image reconstruction, from 2016, include application to magnetic resonance imaging (MRI) [11], with in particular the seminal work of Zhu *et al.* [12], also applied to MRI data. Using deep neural networks for reconstruction of MR images directly from *k*-space data, they also demonstrated preliminary reconstruction results for positron emission tomography (PET) sinogram data. From ∼2018 onward, AI methods exploiting deep networks specifically for PET image reconstruction were increasingly proposed [13], [14]. As would be expected, AI methodology has also been applied to reconstruction in other radiation-imaging modalities, such as CT and SPECT (e.g., [15] and [16]). While this present review will focus on AI for PET reconstruction, many of the approaches are largely also applicable to SPECT, and even CT, thanks to the high

Manuscript received April 30, 2020; revised July 23, 2020; accepted August 1, 2020. Date of publication August 6, 2020; date of current version December 30, 2020. This work was supported in part by the Wellcome/EPSRC Centre for Medical Engineering under Grant WT 203148/Z/16/Z; in part by the King's College London and Imperial College London EPSRC Centre for Doctoral Training in Medical Imaging under Grant EP/L015226/1; and in part by the Centre for Doctoral Training in Medical imaging under Grant EP/S022104/1. *(Corresponding author: Andrew J. Reader.)*

The authors are with the School of Biomedical Engineering and Imaging Sciences, King's College London, London SE1 7EH, U.K. (e-mail: andrew.reader@kcl.ac.uk).

Color versions of one or more of the figures in this article are available online at https://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TRPMS.2020.3014786

flexibility of the mappings that can be trained according to the supplied data in each case.

There have now been a number of reviews on AI, machine learning and deep learning for inverse problems and medical imaging reconstruction (e.g., [1] and [17]–[19]), including potential issues [20]. However, as indicated, this article presents a review of the current state of progress of deep learning within image reconstruction for the specific modality of PET. The format of this article is as follows. Section II reviews the basic principles of conventional or model-based PET image reconstruction. Section III describes the key paradigm shift for PET reconstruction when deep learning is applied, giving a tutorial and overview of deep learning methodology. Section IV briefly overviews four major ways that deep learning can be exploited within PET image reconstruction, and Sections V–VII consider a selection of these in more detail. Finally, Section VIII summarizes the review and offers future perspectives.

II. Basics of Model-Based PET Reconstruction

This section briefly covers the basics of conventional PET image reconstruction, but more comprehensive reviews are of course available (e.g., [21]–[24]).

A. Basic Principles

Image reconstruction for PET involves estimating representation parameters for the spatiotemporal distribution of a radiotracer's concentration in the field of view (FOV) of a PET scanner. For 2-D or 3-D (spatial only) imaging, the model of the tracer distribution $f(\boldsymbol{r})$ is typically a simple linear model parameterized by $\boldsymbol{x}$

$$f(\boldsymbol{r};\boldsymbol{x}) = \sum_{j=1}^{J} x_j b_j(\boldsymbol{r}) \tag{1}$$

where the basis functions $b_j(\boldsymbol{r})$ are usually pixels or voxels, and a parameter vector $\boldsymbol{x} \in \mathbb{R}^J$ specifies the coefficients, or amplitudes, for each basis function $b_j(\boldsymbol{r})$. Throughout this review article the J-dimensional vector $\boldsymbol{x}$ will be taken to represent a 2-D or 3-D reconstructed image, with the assumption that pixels or voxels are used for (1). While the model is nearly always linear, in general it can also be nonlinear, with a key example being consideration of the spatiotemporal (4-D) distribution of the radiotracer, as used in direct reconstruction of radiotracer kinetic parametric maps or 4-D images [25], [26].

With a chosen model of the radiotracer distribution, the next step is to model how the PET scanner would acquire data from this distribution. This concerns modeling the mean of the acquired noisy PET data, based on a given parameter vector $\boldsymbol{x}$. In nearly all cases, a linear model of the data mean is used as follows:

$$\boldsymbol{q}(\boldsymbol{x}) = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{\rho} \tag{2}$$

where $\boldsymbol{A} \in \mathbb{R}^{I\times J}$ is the PET system matrix (also known as the forward model, or system model) and I and J are the number of sinogram bins and the number of voxels of the PET image, respectively, and $\boldsymbol{r}$ is the model of the mean scatter and randoms background. With the object model (1) and the imaging model (2), we then consider the noise model for the data. For PET, the Poisson model is used, as discrete photon counts are recorded

$$m_i \sim \text{Poisson}\{q_i\} \tag{3}$$

where q_i is the model of the mean number of coincidences in the ith line of response (LOR) (or sinogram bin).

Next, it is necessary to define an objective function which indicates how well the parameters $\boldsymbol{x}$ of the model for (1) correspond to the actual measured data, modeled by (2) and (3). The goal of image reconstruction is then to find the parameter vector $\boldsymbol{x}$, for (1), which when forward modeled with (2), best agrees with the acquired noisy measured data (3), according to a chosen objective (or cost) function as follows:

$$\widehat{\boldsymbol{x}} = \underset{\boldsymbol{x}}{\operatorname{argmin}}\, D_{\text{PET}}(\boldsymbol{A}\boldsymbol{x} + \boldsymbol{\rho}; \boldsymbol{m}) \tag{4}$$

where D_{PET} is a function that gives some measure of the distance (discrepancy) between the model of the mean, $\boldsymbol{q}(\boldsymbol{x})$, and the measured data $\boldsymbol{m}$, and so is a measure of data fidelity for any given candidate $\boldsymbol{x}$. For PET, the objective function of choice is the Poisson log likelihood, for which an $\boldsymbol{x}$ should be found which maximizes the likelihood of $\boldsymbol{x}$, given the measured data $\boldsymbol{m}$. When expressed as a distance measure, the negative of the Poisson log likelihood is used (negative, as the Poisson log likelihood needs to be maximized)

$$D_{\text{PET}}(\boldsymbol{q}(\boldsymbol{x}); \boldsymbol{m}) = -\sum_{i=1}^{I} (m_i \ln q_i(\boldsymbol{x}) - q_i(\boldsymbol{x})). \tag{5}$$

A robust way of seeking the extremum of (5) is the maximum likelihood expectation maximization (ML-EM) algorithm [27], [28], where one ML-EM update is given by

$$\boldsymbol{x}^{n+1} = \frac{\boldsymbol{x}^n}{\boldsymbol{A}^T\boldsymbol{1}} \boldsymbol{A}^T \left(\frac{\boldsymbol{m}}{\boldsymbol{A}\boldsymbol{x}^n + \boldsymbol{\rho}} \right) \tag{6}$$

where $\boldsymbol{1} \in \mathbb{R}^I$ and $\boldsymbol{x}^n$ is initialized by uniform values. In (6) (and elsewhere in this article) products and quotients of vectors are element wise, with matrix-vector products using the conventional definition, following the notation introduced by Barrett *et al.* [29].

B. Regularization by Analysis/Encoding

Since the measured data are noisy, minimizing (5) [e.g., through use of (6)] results in typically noisy estimates of the radiotracer distribution via (1), as most often voxel basis functions are chosen. For very noisy data, "night sky" reconstructions are obtained. Therefore, regularization is used to seek noise-compensated representations of the radiotracer distribution. This is usually achieved by including a penalty term $R(\boldsymbol{x})$ in the objective function

$$\widehat{\boldsymbol{x}} = \underset{\boldsymbol{x}}{\operatorname{argmin}}\, D_{\text{PET}}(\boldsymbol{q}(\boldsymbol{x}); \boldsymbol{m}) + \beta R(\boldsymbol{x}) \tag{7}$$

where the hyperparameter β controls the strength of regularization relative to fidelity to the measured data. The penalty

term $R(\boldsymbol{x})$ can be any of a wide range of priors, designed to encourage solutions which agree with our prior belief regarding the radiotracer distribution. If $\boldsymbol{x}$ does not agree well with our prior belief, $R(\boldsymbol{x})$ tends to be large, and vice versa. A common choice is to expect the neighboring voxel values in $\boldsymbol{x}$ to be similar, so that $R(\boldsymbol{x})$ is some function of the voxel-value differences between neighboring voxels. A common example is

$$R(\boldsymbol{x}) = \frac{1}{4}\sum_{j=1}^{J}\sum_{l=1}^{J} w_{jl}\phi\left(x_j - x_l\right) \tag{8}$$

where $\phi(.)$ is a potential function, such as a quadratic [for which the helpful normalization of 1/4 is already placed in (8)], so that any differences between voxel values result in an increased value of R, thereby penalizing choices of $\boldsymbol{x}$ which have largely varying neighboring values, often the result of fitting closely to the noise in the data $\boldsymbol{m}$. The weights ($\boldsymbol{w} \in \mathbb{R}^{J\times J}$, although usually limited to a small patch neighborhood) allow guidance from anatomical images such as MRI [30]. We make an advance observation that, in the context of what will follow later in this review, priors such as (8) are mathematically convenient, or handcrafted/designed priors, and not directly evidence or data-based. To build a more general version of (8), the following vector can be considered:

$$\boldsymbol{z} = \phi(\boldsymbol{H}x) \tag{9}$$

where $\boldsymbol{H} \in \mathbb{R}^{J\times J}$ is a matrix, which would be a finite difference operator to mimic (8), and $\boldsymbol{z}$ is some "coded" representation of $\boldsymbol{x}$ obtained by the overall transform, and then

$$R(\boldsymbol{x}) = \mathbf{1}^T\boldsymbol{z} \tag{10}$$

where $\mathbf{1} \in \mathbb{R}^J$, to achieve a summation of the contents of $\boldsymbol{z}$.

The approach to regularization given by (7), with the example of (8), can be referred to as *analysis* regularization. Effectively any candidate object representation $\boldsymbol{x}$ is *analysed* by being transformed by an operator (such as $\boldsymbol{H}$, followed by ϕ), whereby the operator or transform is designed such that the output $\boldsymbol{z}$ should be small valued for candidate $\boldsymbol{x}$ solutions which agree with our prior beliefs. Here, "small valued" means that the sum of $\boldsymbol{z}$ should be small, which can be achieved, for example, by $\boldsymbol{z}$ being sparse (i.e., only a limited number of nonzero elements). Hence, if $\boldsymbol{H}$ is a gradient operator, or, as another example, a wavelet transform, then solutions of $\boldsymbol{x}$ which have limited gradients (e.g., piecewise smooth objects), or limited wavelet coefficients (e.g., images which are readily compressible) are encouraged, respectively. In the latter case, it can be noted that natural and noise-free images are more readily compressed than noise-ridden images. This approach is used within compressed sensing methods in MRI [31], where the reconstructed image is required to be sparse in some transform domain, a strongly informative regularization which permits fewer k-space samples to be acquired.

Analysis regularization can be achieved in PET imaging using an MAP-EM algorithm, such as that of De Pierro [32], which is a convergent algorithm for priors such as (8), provided that the potential function $\phi(.)$ is convex. The iterative update of an image estimate $\boldsymbol{x}^n$, when the prior is of the form of (8) with a quadratic potential function is

$$x_j^{n+1} = \frac{2x_j^{EM}}{\left(1-\beta v_j x_j^{SM}\right) + \sqrt{\left(1-\beta v_j x_j^{SM}\right)^2 + 4\beta v_j x_j^{EM}}} \tag{11}$$

where $\boldsymbol{x}^{EM}$ corresponds to the ML-EM update of $\boldsymbol{x}^n$ (6), $\boldsymbol{s} = \boldsymbol{A}^T\mathbf{1}$ (the sensitivity image) and

$$v_j = \frac{\sum_{l=1}^{J} w_{jl}}{s_j} \tag{12}$$

with

$$x_j^{SM} = \frac{1}{2\sum_{l=1}^{J} w_{jl}}\sum_{l=1}^{J} w_{jl}\left(x_j^n + x_l^n\right) \tag{13}$$

being effectively a weighted, potentially edge-constrained, smooth of the current estimate $\boldsymbol{x}^n$. Note that (13) does not explicitly contain the potential function as a quadratic potential has been used in this example, based on the update from [33].

To finish this brief review of analysis regularization, one more important case worth mentioning in the context of conventional PET reconstruction is the simple case of using a prior image for a quadratic penalty

$$R(\boldsymbol{x}) = \sum_{j=1}^{J}\left(p_j - x_j\right)^2 \tag{14}$$

where $\boldsymbol{p}$ is a prior image from which the estimate of $\boldsymbol{x}$ should not deviate too far. Whilst proposed very early on by Levitan and Herman for MAP-EM reconstruction [34], and while not at all frequently used in conventional PET reconstruction, this analysis regularization method has however found great utility when deep learning is applied to PET reconstruction, as will be discussed later. Using the penalty of (14), an iterative update of $\boldsymbol{x}^n$ can be found by a simple combination of the prior image $\boldsymbol{p}$ and the standard EM update image [found from (6)]

$$x_j^{n+1} = \frac{2s_j x_j^{EM}}{\left(s_j - \beta p_j\right) + \sqrt{\left(s_j - \beta p_j\right)^2 + 4\beta x_j^{EM} s_j}} \tag{15}$$

where similarity to the update of (11) is notable, with equivalence arising only if $\sum_{l=1}^{J} w_{jl} = 1$ and $\boldsymbol{x}^{SM} = \boldsymbol{p}$.

C. Regularization by Synthesis/Generators

A second major way of introducing our prior expectations about what $\boldsymbol{x}$ should look like is to instead express $\boldsymbol{x}$ as the output of some operator, where the operator is designed so as to only generate candidate $\boldsymbol{x}$ vectors which agree with our prior beliefs. A simple linear example is to use a matrix containing basis vectors, such that the output $\boldsymbol{x}$ is synthesized by summation of these basis vectors

$$\boldsymbol{x} = \boldsymbol{B}\boldsymbol{z} \tag{16}$$

where in this context $\boldsymbol{z}$ is now a vector of coefficients, which can be viewed as a coded or latent representation of $\boldsymbol{x}$. The matrix of basis vectors, $\boldsymbol{B}$, can also be referred to as a dictionary containing atoms. We can achieve regularizing constraints

on the output $\boldsymbol{x}$ in three main ways: 1) enforcing non-negative values for $\boldsymbol{z}$ (crucial if $\boldsymbol{B}$ is full rank); 2) explicitly using a reduced set of basis vectors in $\boldsymbol{B}$, by limiting the dimensions of $\boldsymbol{z}$ to be smaller than the dimensions of $\boldsymbol{x}$; or 3) using a complete set of basis vectors in $\boldsymbol{B}$, or even an overcomplete dictionary of basis vectors whilst requiring $\boldsymbol{z}$ to be a sparse vector (e.g., by use of a norm of $\boldsymbol{z}$ as a penalty). The first approach is the most simple, and has been used in PET, as the popular ML-EM method of (6) naturally gives non-negative solution vectors. Hence, ML-EM can be rewritten to directly estimate the latent code (coefficients) vector $\boldsymbol{z}$

$$z^{n+1} = \frac{z^n}{\boldsymbol{B}^T\boldsymbol{A}^T\mathbf{1}}\boldsymbol{B}^T\boldsymbol{A}^T\left(\frac{\boldsymbol{m}}{\boldsymbol{A}\boldsymbol{B}z^n + \boldsymbol{\rho}}\right) \tag{17}$$

with the final reconstruction given by (16). Example choices for $\boldsymbol{B}$ include MR-derived basis functions based on similarity between MR voxel values, or ones derived by time-activity curve (TAC) similarity between voxels, found by the kernel method. Hence, (17) with an image model of (16) is often called kernel EM (KEM) [35]–[37]. Any positive-valued $\boldsymbol{z}$ vector will always deliver an image of positive-valued weighted sets of MR-anatomy or TAC inspired basis vectors/dictionary atoms, eliminating the possibility of noisy night sky reconstructions. A purely temporal version of (17) for 4-D PET reconstruction [38], involves alternating estimation of not only $\boldsymbol{z}$ but also estimation of a compressed, limited-dimensional, $\boldsymbol{B}$.

D. Drawbacks

We now observe three potentially undesirable aspects with the aforementioned conventional model-based approaches to PET image reconstruction.

1) Noisy Data: Since the data are noisy, choosing to fit parameter estimates $\mathbf{x}$ to noisy data $\mathbf{m}$ yields noisy reconstructed images, suggesting that even the very starting point of a data-fidelity objective function such as (4) is not really what is desired.

2) Need for Regularization: Compensating for the first problem by regularization with a function $R(\boldsymbol{x})$ [as in (7)] involves user-specified/hand-crafted prior assumptions [such as (8)], in terms of what is, and what is not, acceptable for the image properties. Even if we do have a good prior, how strong should it be (β) in comparison to data fidelity? How can we make such selections? This is an active area of research (e.g., [39] and [40]). Also, regularization by means of synthesis/basis function methods usually involves similar suboptimal user-specified representations, with comparable issues of hyperparameter selection.

3) Modeling Assumptions: The methods described all presuppose accurate and precise knowledge of the model of the mean of the data, through the forward model matrix $\boldsymbol{A}$, and also knowledge of the noise distribution of the data vector $\boldsymbol{m}$.

All of these potential concerns can be addressed by the use of AI, or more specifically deep learning, for PET image reconstruction. For reference, the conventional model-based approach to image reconstruction, as outlined in this section, is shown schematically in Fig. 1.

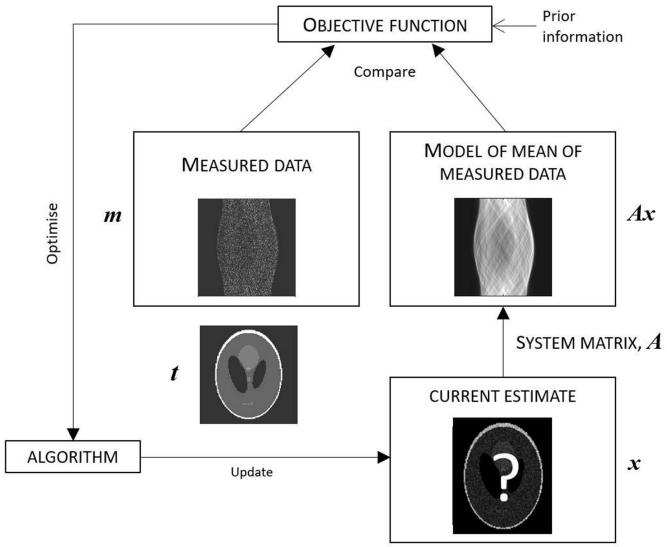

Fig. 1. Work flow overview for conventional model-based PET image reconstruction. Note that explicit consideration of the ground truth $\boldsymbol{t}$ does not enter into the process at any point. This omission is the key reason why AI is able to offer a radically different approach, by making use of either the actual ground truth (e.g., via simulations) or an estimate of the truth (e.g., higher-count reference data).

III. AI Paradigm Shift and Theory

This section now considers the key paradigm shift when using AI approaches for PET image reconstruction, and reviews methodology for direct deep-learning reconstruction from PET data. A key concept is that of *learning* how to reconstruct a high quality image from a noisy dataset, through the use of *training data*.

A. Basic Principles

The AI approach is a fundamental shift in focus in comparison to the conventional model-based framework outlined in the previous section. In broad terms the key is this: we no longer define the noisy measured data $\boldsymbol{m}$ as the target in the objective function (4), but instead we use a high quality desirable reference $\boldsymbol{t}$ as the target in a new objective function. Thus, instead of fitting parameters $\boldsymbol{x}$ to noisy data $\boldsymbol{m}$, and then trying to compensate for noise in the data by $R(\boldsymbol{x})$, with an AI approach we instead choose to *estimate a mapping, F,* that takes us from $\boldsymbol{m}$ to an estimate of $\boldsymbol{x}$ corresponding to what we would actually want. Ideally, we would want the ground truth radiotracer distribution $\boldsymbol{t}$ that had given rise to $\boldsymbol{m}$, or, lacking that, a very high-statistical quality reconstructed image.

This is achieved by *learning* a reconstruction operator, or mapping, F, using example *training data*. The mapping is parameterized by a vector $\boldsymbol{\theta}$, which would ideally take us directly from the noisy data $\boldsymbol{m}$ to what we desire, $\boldsymbol{t}$. Such a mapping would implicitly need to account for the entire physics of the imaging process and the noise distribution of the data, all within F. Of course, the mapping will also need to generalize well for any new dataset $\boldsymbol{m}$, being able to map an unseen dataset to the unknown ground truth. This places importance on the training data being diverse and extensive

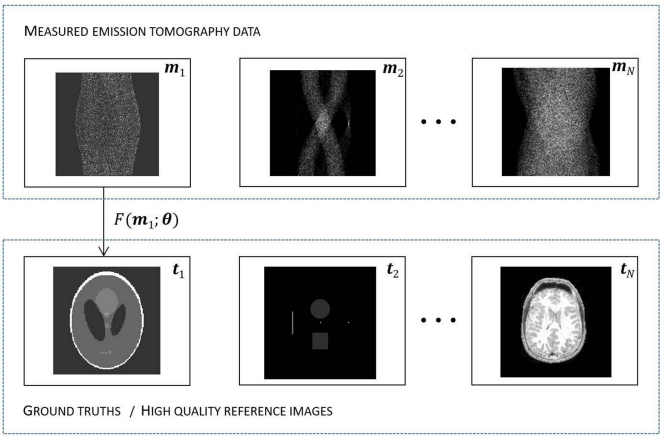

Fig. 2. AI paradigm for direct PET reconstruction: we find (or *learn*) one mapping F which maps each data vector $\mathbf{m}$ to a desirable target vector $\mathbf{t}$. Supervised learning of the mapping needs example pairs of inputs and expected outputs (called targets or labels), which form the training data for the learning process. More advanced methods [4] learn how to map from one distribution to the other distribution, evading the need for paired data vectors.

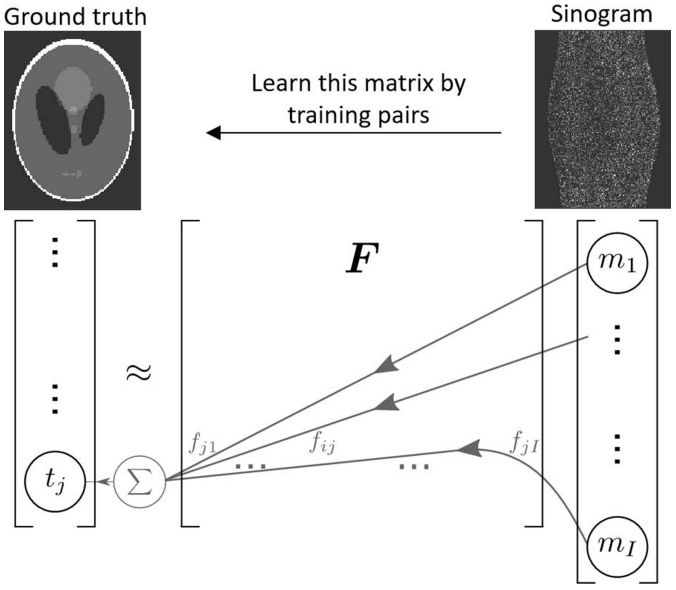

Fig. 3. Direct linear mapping approach. *Top:* the matrix $\boldsymbol{F}$ is trained to map data $\boldsymbol{m}$ to the ground truth or reference $\boldsymbol{t}$. *Bottom:* when the trained $\boldsymbol{F}$ is presented with a new dataset $\boldsymbol{m}$, a given output value is obtained by a weighted sum of the input vector elements in $\mathbf{m}$, where the weights for a given output element i are contained along a row i of the matrix $\boldsymbol{F}$. This reveals the link to neural networks, for which the above case is termed a FC layer (which in general allow a bias to be added to each output, with subsequent optional application of a nonlinear function).

enough to adequately represent the domain of possible future input datasets.

During training of the mapping it is of course unreasonable to expect to find a single mapping that will always directly deliver the ground truth $\boldsymbol{t}$ from a given noisy input $\boldsymbol{m}$, and so an objective function is required, which seeks to match the mapping of $\boldsymbol{m}$ (through F) to the target $\boldsymbol{t}$ as closely as possible to within some tolerance, or *loss*.

In the context of deep learning, the parameters $\boldsymbol{\theta}$ of the mapping are optimized so as to minimize a *loss function*, given by

$$\widehat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}}{\operatorname{argmin}} \sum_{n=1}^{N} D_{\mathrm{NET}}(F(\boldsymbol{m}_n;\boldsymbol{\theta});\boldsymbol{t}_n) \tag{18}$$

where the mapping F is parameterized in some way by a vector of parameters $\boldsymbol{\theta}$, such that when the mapping is applied to one of the $n = 1 \cdots N$ input datasets in the training data, e.g., $\boldsymbol{m}_n$, the mapping generates an output which should be close to $\boldsymbol{t}_n$ (see Fig. 2). The key aspect to the loss function D_{NET} for the mapping (often a network) is that it needs to be defined over many such example training dataset pairs (inputs $\mathbf{m}_n$, each paired with desired outputs $\boldsymbol{t}_n$) that adequately cover the domain of potential future inputs. This means the training seeks just one single mapping F, which will best fit each and every example training noisy dataset $\boldsymbol{m}_n$ to its corresponding high quality reference $\boldsymbol{t}_n$. During training, often a separate validation dataset is used to monitor performance for data unseen by the optimization. For example, if the loss function, when evaluated on the validation data, starts to increase, this is indicative of overfitting to the training data, and so the training process can be halted.

The expectation, when training is complete, is that a new supplied input measured dataset $\boldsymbol{m}$ will be mapped using F to predict the unknown ground truth for the new dataset

$$\widehat{\boldsymbol{x}} = F\big(\boldsymbol{m};\widehat{\boldsymbol{\theta}}\big). \tag{19}$$

We expect therefore generalization to unseen, future data, on the assumption that the unseen data comes from the same domain as the training data. The challenge of dealing with new data that is outside the domain of the training data is known as domain adaptation, an active area of research [41].

B. Linear Direct Mapping: Fully Connected Layer

The simplest case would be to find a purely linear mapping

$$\widehat{\boldsymbol{x}} = \boldsymbol{F}_{\widehat{\boldsymbol{\theta}}_\ell}\boldsymbol{m} \tag{20}$$

where the mapping F is now just a matrix $\boldsymbol{F} \in \mathbb{R}^{J\times I}$, (see Fig. 3) and we have added a subscript $\widehat{\boldsymbol{\theta}}_\ell$ to denote that this matrix depends on the trained parameter vector. The extra subscript ℓ denotes what we will refer to from now on as a *layer*, described further below. This matrix mapping can be regarded as a *single layer* network—whereby each output value $\hat{x}_i$ is just a weighted sum of the input values in $\boldsymbol{m}$, with the weights (neurons) given by the ith row of matrix $\boldsymbol{F}$. (As a brief aside, we note that a simple nonlinear function can, optionally, be applied to each output element in the vector—this will be considered further below). It may seem like an ambitious task to estimate, and we can see that we would likely need many training pairs of $\boldsymbol{m}$ and $\boldsymbol{t}$ in order to find an $\boldsymbol{F}$ that will be able to generalize for unseen input vectors $\boldsymbol{m}$. In fact, for the typical scale of 2-D and 3-D PET image reconstruction, we would need to estimate anywhere from millions to trillions of parameters! But given, for example, the existence of linear PET image reconstruction methods, such as filtered backprojection (FBP) [42], [43], backproject then filter (BPF) [44], or better still the Moore–Penrose pseudo inverse via singular value decomposition (SVD) [45], [46], it is evident that

linear mappings do exist that can achieve good quality reconstructions of $\boldsymbol{x}$ from $\boldsymbol{m}$. Likewise in MR, the default inverse Fourier transform is a good starting point for a linear operator. The advantage, again, is that a learned reconstruction operator would not only account for the imaging physics but would implicitly also include a data-trained noise-reduction strategy. This is in contrast to FBP (where an empirically chosen filter cut off is needed), or in contrast to a pseudoinverse (where the modulation or truncation of the inverse of the singular value spectrum is similarly empirically chosen to compensate for noise).

In the context of deep learning, a matrix like that in (20) is known as a fully connected layer (FC layer), or a *dense layer* (since every single input value can affect every single output value). The use of the word layer (inspired by the neuroanatomy of the cerebral cortex) arises from the fact that, as we will see later, we may use more than one single mapping in a sequence—we can cascade a series of mappings more generally. Each extra mapping is a layer, and when we have multiple layers we have a *deep* network, hence the term deep neural network. The use of such multiple layers typically arises when a nonlinear function is introduced between layers, but even using a series of purely linear operators certainly is not trivial, as will be discussed later.

A final note here is to mention that some conventions refer to the input or output of a given single operator as a layer. However, here (similar to [47] and [48]) we use the word layer to refer to the operator itself, but according to the context, one can loosely use the word layer in reference to the output of the operator as well.

C. Convolution Direct Mapping: Convolutional Layer

Before considering more complex mappings, we will now consider a simple but very instructive example of a mapping that is not only linear but also *shift-invariant*—convolution. To motivate this simple example, we will now consider the input measured data $\boldsymbol{m}$ to be purely a noisy version of $\boldsymbol{t}$ [i.e., using modeling (1)–(3), but now taking $\boldsymbol{A} = \boldsymbol{I}$, so that no sinogram is now needed]. We will then seek a single convolution kernel, such that when convolved with the noisy data $\boldsymbol{m}$ (which is now regarded as a noisy image in this instance), gives a best fit to the high quality reference $\boldsymbol{t}$. This could be written as (20), with the requirement that $\boldsymbol{F}$ now be a circulant matrix (i.e., achieving convolution). More explicitly

$$\widehat{\boldsymbol{x}} = \boldsymbol{C}_{\widehat{\boldsymbol{\theta}}_\ell}\boldsymbol{m} \tag{21}$$

where the circulant matrix $\boldsymbol{C}$ contains a unique 2-D or 3-D kernel, defined by parameters $\widehat{\boldsymbol{\theta}}_\ell$, such that the kernel is duplicated and shifted in successive columns of the matrix $\boldsymbol{C}$. This mapping would correspond to a layer which is called a *convolutional layer*, with, in this simple case, just one kernel.

Fig. 4 illustrates the capabilities of learning just one single convolution kernel for a purely linear and shift-invariant (LSI) mapping. Results are shown for optimizing the parameters of a single kernel for two different example applications. The first is to denoise, i.e., to match $\boldsymbol{m}$ to $\boldsymbol{t}$ using a least-squares loss function [usually referred to as the mean square error (MSE) loss function in the machine learning literature]. The results, in this case are as expected—for noisier input measured data, a broader trained kernel is obtained in order to achieve denoising, and for less noisy data a narrower kernel is obtained, as less denoising is required. The second example is using a kernel to sharpen an image, removal of blurring—again, the results show optimization of the kernel to be effective in deblurring.

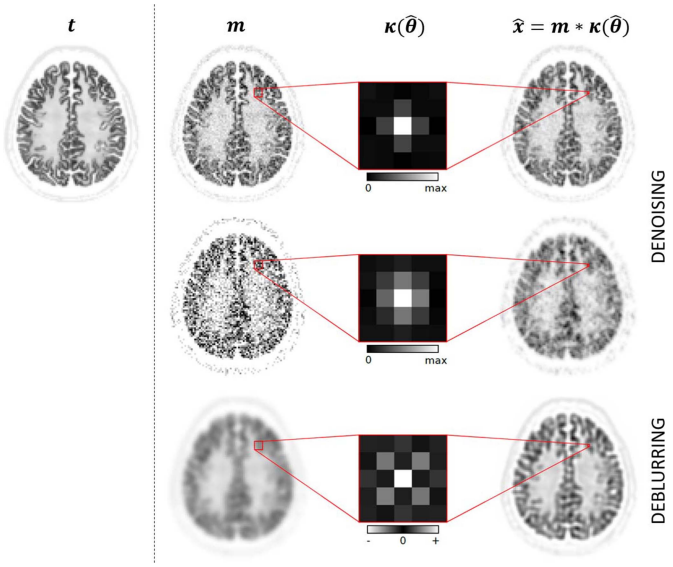

Fig. 4. Three examples of a direct convolution (linear shift-invariant) mapping approach, with data-driven learning of a single kernel to try and map $\boldsymbol{m}$ to $\boldsymbol{t}$. For noisier data (row 2), more neighborhood averaging is needed to denoise, and so a broader kernel was learned. For the different task of deconvolution (row 3), a sharpening kernel was learned, to try and match $\boldsymbol{t}$.

D. Convolution With Nonlinearity: Feature Maps

The convolution mapping can be extended to include a nonlinearity afterwards, which can be sometimes regarded as a separate layer. The nonlinearity is simply application of a nonlinear function, which we will call $\sigma(.)$, element by element, on each pixel or voxel value of the output vector of the convolution (where the output is often referred to as a *feature map*). It is called an *activation function*, as it often serves to suppress values in the output, and let others pass through (as activated values). So, for a single layer convolutional network we would have

$$\widehat{\boldsymbol{x}} = \sigma_\ell\left(\boldsymbol{C}_{\widehat{\boldsymbol{\theta}}_\ell}\boldsymbol{m}\right). \tag{22}$$

If we choose $\sigma_\ell(.)$ to be a rectified linear unit (ReLU) [49], it sets any negative values to zero, retaining all positive values as they are. Fig. 5 shows examples of the utility of applying this nonlinearity, which in the figure is shown in the thresholded column. The thresholded feature maps show, for example, edges, or a tumour location. Hence, the nonlinearity can give even more useful feature maps of specific interest, with background aspects removed. For the case of ReLU, the nonlinearity amounts to thresholding, deleting background information, and keeping desired features. Many other nonlinear activation functions exist, including, for example, leaky

ReLU (LReLU, attenuating rather than removing negative values), sigmoid (for constraining outputs to be from 0 to 1) and hyperbolic tangent (tanh, for constraining outputs to be from -1 to $+1$). Furthermore, the use of an offset or scalar bias value b just prior to an activation such as ReLU, allows adjustment of the level of thresholding without changing the activation function $\sigma(.)$

$$\widehat{\boldsymbol{x}} = \sigma_\ell\left(\boldsymbol{C}_{\widehat{\boldsymbol{\theta}}_\ell}\boldsymbol{m} + b_{\widehat{\boldsymbol{\theta}}_\ell}\right) \tag{23}$$

where the bias is a single trainable offset scalar parameter, the single value of which is now included into the overall vector of parameters for the layer, $\widehat{\boldsymbol{\theta}}_\ell$.

Given the utility of convolution with a bias and activation for delivering a feature map, we note that for a given input image, it would be useful to obtain more than just one single feature map. This is already shown in Fig. 5, where we have 3 different feature maps arising from one image. This involves generating more than one output, by *using more than one kernel in a convolutional layer*. So starting from (23) we can use multiple kernels in this one single convolutional layer, to obtain multiple outputs—one for each kernel

$$\widehat{\boldsymbol{x}} = \sigma_\ell\left(\begin{bmatrix} \boldsymbol{C}^1_{\widehat{\boldsymbol{\theta}}_\ell} \\ \vdots \\ \boldsymbol{C}^K_{\widehat{\boldsymbol{\theta}}_\ell} \end{bmatrix}\boldsymbol{m} + \boldsymbol{b}_{\widehat{\boldsymbol{\theta}}_\ell}\right) \tag{24}$$

where the output vector $\widehat{\boldsymbol{x}}$ is now K times larger (i.e., there are as many output images as there are kernels). This corresponds to the number, $k = 1 \cdots K$, of convolution matrices applied to $\boldsymbol{m}$. Note further that we have a unique scalar bias value for each kernel, represented in (24) by a single vector $\boldsymbol{b}$, and that this vector and each of the kernels all depend on the overall set of parameters, $\widehat{\boldsymbol{\theta}}_\ell$, for this layer.

Fig. 5 illustrates (24) for the choice of just three kernels—which in this figure are purely handcrafted to show the flexibility of different kernels. In deep learning however, the kernels are randomly initialized, and the training process adapts the kernel values to obtain the feature maps necessary to make the outputs ultimately serve to match the target (i.e. to minimize the loss function).

Finally, we note that often we require just one image output, whereas using multiple kernels in a layer delivers multiple outputs. These multiple outputs are referred to as channels. We can easily join these channels together into one single output by applying one more convolutional layer, with just one kernel (to give just one output), but with the single kernel having multiple channels (one for each of the input channels to the layer). This adds together the feature maps as follows:

$$\widehat{\boldsymbol{x}} = \begin{bmatrix} \boldsymbol{C}^{1,1}_{\widehat{\boldsymbol{\theta}}_{\ell+1}} & \cdots & \boldsymbol{C}^{1,K}_{\widehat{\boldsymbol{\theta}}_{\ell+1}} \end{bmatrix}\sigma_\ell\left(\begin{bmatrix} \boldsymbol{C}^{1,1}_{\widehat{\boldsymbol{\theta}}_\ell} \\ \vdots \\ \boldsymbol{C}^{K,1}_{\widehat{\boldsymbol{\theta}}_\ell} \end{bmatrix}\boldsymbol{m} + \boldsymbol{b}_{\widehat{\boldsymbol{\theta}}_\ell}\right) \tag{25}$$

where we use a second superscript to denote the channel number, with the first superscript kept for the kernel number. There are as many channels for this one kernel in this second layer (labelled $\ell + 1$) as there are kernels (hence outputs) from the first layer (labelled ℓ). A simple choice is to use single pixel or voxel kernels for each of the channels of the single kernel, so that the final output is just a weighted sum of the feature maps from the previous layer (as was illustrated in Fig. 5).

E. Deep Networks: Mappings With Multiple Layers

We now more explicitly consider using a series, or cascade, of mappings, to form a deep network. To start with, we could take the purely linear mapping of (20) as a series of matrix operators, a series of layers, each layer defined by a set of parameter values, so for $\ell = 1 \cdots L$ FC layers we would have

$$\widehat{\boldsymbol{x}} = \boldsymbol{F}_{\widehat{\boldsymbol{\theta}}_L} \cdots \boldsymbol{F}_{\widehat{\boldsymbol{\theta}}_\ell} \cdots \boldsymbol{F}_{\widehat{\boldsymbol{\theta}}_2}\boldsymbol{F}_{\widehat{\boldsymbol{\theta}}_1}\boldsymbol{m} \tag{26}$$

forming a deep neural network, with the overall complete set of parameters of the mapping given by $\boldsymbol{\theta}$, composed of all the parameters for each of the layers. More generally we have, at a given layer ℓ, an intermediate, *latent* or *hidden* vector of results (i.e., not visible at the input or output), z_ℓ, given by

$$z_\ell = \boldsymbol{F}_{\widehat{\boldsymbol{\theta}}_\ell} z_{\ell-1} \tag{27}$$

for $\ell = 1 \cdots L$, with the final output being $\widehat{\boldsymbol{x}} = z_L$ and with the first input being $z_0 = \boldsymbol{m}$. Construction of the number and size of these mappings refers to the *architecture* of the network. At first sight, for such a purely linear model, it can seem that (20) and (26) are completely equivalent when appropriate choices of parameters are made. However, the precise architecture does profoundly matter (e.g., sizes of matrices used), in terms of model constraints, number of summations and products involved, and ease of training of the parameters. As a first example, we could use the form of (26) to learn a diagonalization of a linear mapping which is comparable to a truncated version of the SVD-inverse, using a series of 3 matrix layers. A further illustrative example is that of the discrete Fourier transform (DFT), which can also be represented by (20), whereas a linear rearrangement of this purely linear transform into a fast Fourier transform (FFT), which could be written as (26), has a profound impact on processing speed.

Of course, convolutions can also be cascaded into a series, increasing depth of the mapping. We could have

$$\widehat{\boldsymbol{x}} = \boldsymbol{C}_{\widehat{\boldsymbol{\theta}}_L} \cdots \boldsymbol{C}_{\widehat{\boldsymbol{\theta}}_\ell} \cdots \boldsymbol{C}_{\widehat{\boldsymbol{\theta}}_2}\boldsymbol{C}_{\widehat{\boldsymbol{\theta}}_1}\boldsymbol{m}. \tag{28}$$

Just as (26) was not trivial due to the capability of varying the size of the matrices in the series of layers, so also (28) should not be regarded as trivial—it is possible to use stride to vary the size of the latent output vector at each layer (the feature map sizes), thereby imposing constraints on the model, as will be discussed further below. We note here that such a series of convolutions allows the concept of *receptive field* to be understood—a pixel or voxel in the input $\mathbf{m}$ can reach or affect a voxel in the output image or map, according to the overall reach of successive application of a series of convolution kernels. So rewriting (27) for convolution, generally, at a given layer ℓ we would have

$$z_\ell = \boldsymbol{C}_{\widehat{\boldsymbol{\theta}}_\ell} z_{\ell-1}. \tag{29}$$

Recalling that we might also generate more than one output feature map at a given layer ℓ by using more than one kernel

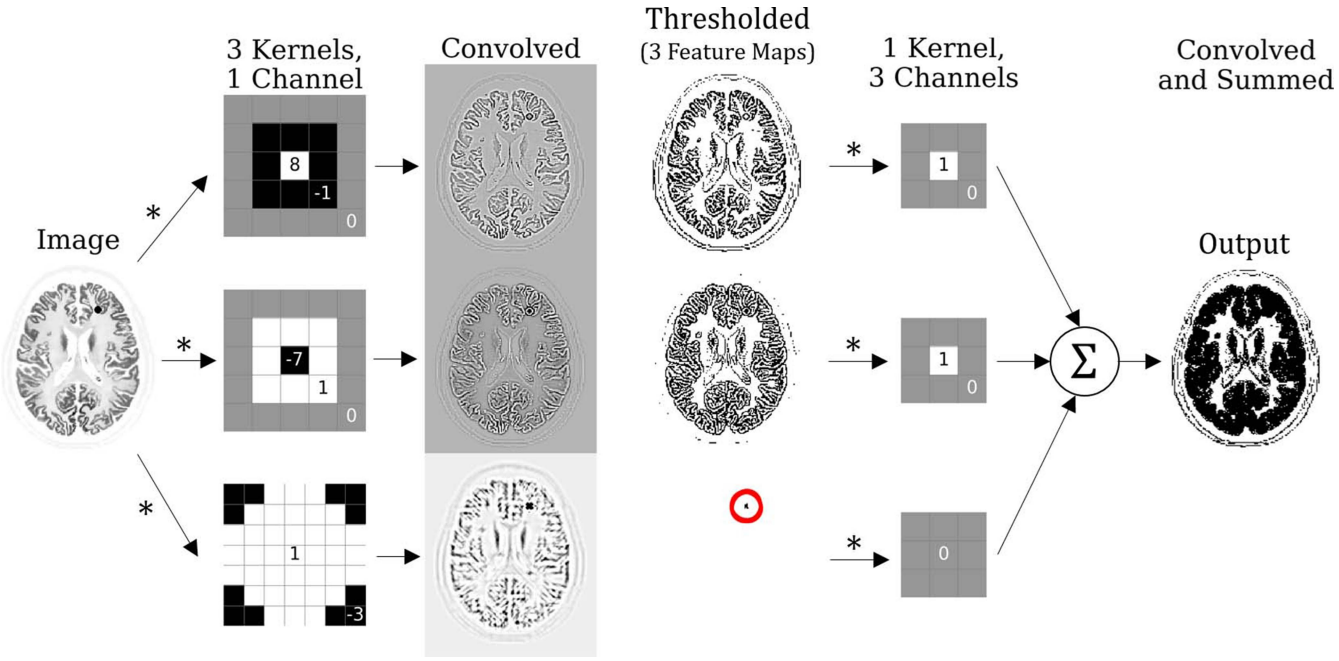

Fig. 5. Demonstration of a two-layer convolutional mapping, with a nonlinearity included (nonlinear shift-invariant). The input image is convolved with three different kernels (each having just one channel, as there is only one input image), giving three convolved outputs. Each convolution output is then, in effect, thresholded by adding some positive or negative offset (a *bias*) (a single unique value for each convolved output), then setting negatives to zero (e.g., by a ReLU activation). These three resulting *feature maps* are then summed with differing weights to deliver a final output image, where in the example shown a 3-channel kernel is used to achieve this, with the last channel being the example of a zero kernel which can remove that feature. The number of inputs to a convolutional layer determines the number of channels needed by a kernel in the layer, and the number of kernels used determines the number of outputs from the layer. The brain phantom used in this example (and in Figs. 5, 7, and 8) is from [104].

in a convolutional layer, we can write

$$z_\ell = \begin{bmatrix} \boldsymbol{C}^1_{\widehat{\boldsymbol{\theta}}_\ell} \\ \vdots \\ \boldsymbol{C}^K_{\widehat{\boldsymbol{\theta}}_\ell} \end{bmatrix} z \Bigg|_{\ell-1} \tag{30}$$

where the output z_ℓ would now be K times the size of the input vector $z_{\ell-1}$, according to the number, $k = 1 \cdots K$, of convolution matrices applied to z_ℓ. If we use the model of (30), then to add on another convolutional layer, it will need to operate on more than one output image—we will have a multichannel input $z_{\ell-1}$, and so we need a kernel for each of these inputs present in $z_{\ell-1}$—this gives rise to the need for a *multichannel kernel*. Combining the idea of multiple kernels, with each being multichannel, we have the overall LSI mapping of

$$\boldsymbol{W}_{\widehat{\boldsymbol{\theta}}_\ell} = \begin{bmatrix} \boldsymbol{C}^{1,1}_{\widehat{\boldsymbol{\theta}}_\ell} & \cdots & \boldsymbol{C}^{1,C}_{\widehat{\boldsymbol{\theta}}_\ell} \\ \vdots & \ddots & \vdots \\ \boldsymbol{C}^{K,1}_{\widehat{\boldsymbol{\theta}}_\ell} & \cdots & \boldsymbol{C}^{K,C}_{\widehat{\boldsymbol{\theta}}_\ell} \end{bmatrix} \tag{31}$$

for channels $c = 1 \cdots C$, and kernels $k = 1 \cdots K$, at layer ℓ. It is easy to note from (31) that when using multichannel kernels, the mapping adds together the outputs for each channel of the kernel, so that again each single kernel, whether multichannel or not, still gives just one output image to present to the next layer.

We finish this section by noting that we can cascade these multikernel, multichannel mappings

$$\widehat{\boldsymbol{x}} = \boldsymbol{W}_{\widehat{\boldsymbol{\theta}}_L} \cdots \boldsymbol{W}_{\widehat{\boldsymbol{\theta}}_\ell} \cdots \boldsymbol{W}_{\widehat{\boldsymbol{\theta}}_2} \boldsymbol{W}_{\widehat{\boldsymbol{\theta}}_1} \boldsymbol{m} \tag{32}$$

and since we often desire the output size of $\boldsymbol{x}$ to be a single image, usually the very last layer $\boldsymbol{W}_{\widehat{\boldsymbol{\theta}}_L}$ is a multichannel single kernel (e.g., with each kernel being just a delta function). This is in order to synthesize a single output image from multiple input feature maps from the penultimate layer, equivalent to just a weighted sum of these input channels, where the weights are learned.

A simple linear *autoencoder* [50] can take the form of (32), by downsampling in early layers (i.e., using a convolution stride greater than 1) and increasing the number of kernels (feature maps), and then upsampling in later layers (through use of fractional stride) and reducing the number of feature maps. In the context of an autoencoder, the goal is to match the input and output, requiring the mapping to pass through a latent space bottleneck [e.g., as a midpoint layer in (32)] which should capture features (high feature dimension via a high number of kernels) with only limited spatial information (high level of downsampling). Noise is unlikely to be represented in this compressed latent representation space. This will be considered further later in this section when the case of a convolutional encoder–decoder mapping is covered.

F. Convolutional Neural Networks

The multilayer convolution mapping can be extended to include nonlinearities between layers. First, a general layer can be given by

$$z_\ell = \sigma_\ell \left(\boldsymbol{W}_{\widehat{\boldsymbol{\theta}}_\ell} z_{\ell-1} + \boldsymbol{b}_\ell \right) \tag{33}$$

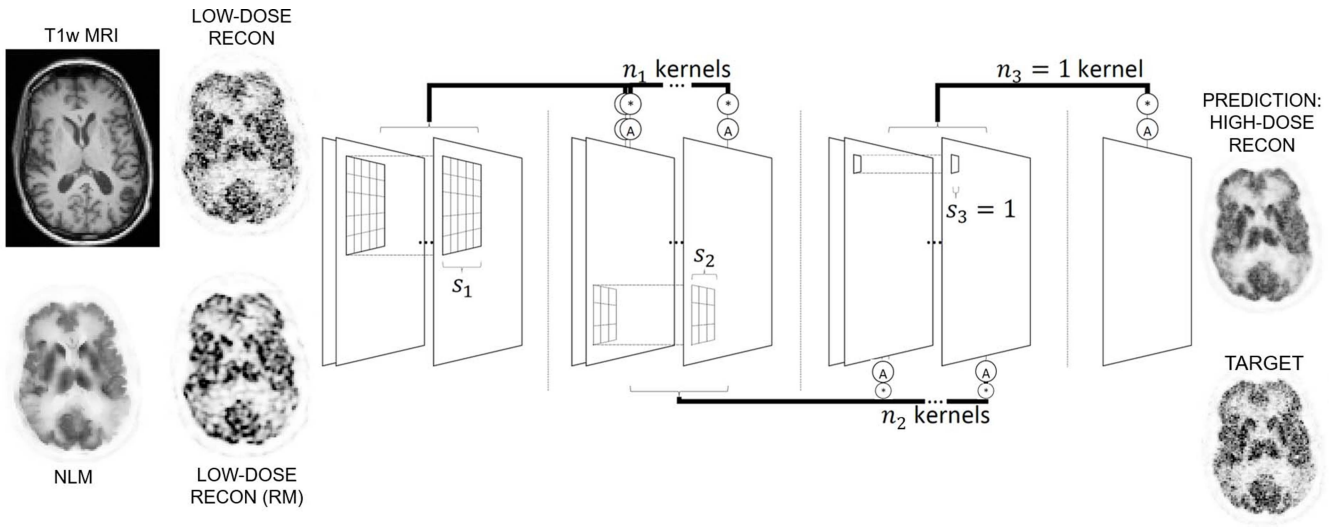

Fig. 6. Example CNN, based on [55], composed of three convolutional layers, designed to map low-dose PET images to full-dose PET images. Here, there are 4 input channel images: a T1-weighted MR, 2 different PET reconstructions of the same low-dose PET data [one with, one without resolution modeling (RM)], and a post-processed PET reconstruction (nonlocal means). These 4 inputs are fed into three convolutional layers. The final convolutional layer is composed of a multichannel single kernel, to synthesize just one output image intended to predict the full-dose PET image.

where the bias is such that it is a single trainable offset scalar parameter for each kernel, such that the argument of the activation function in (33) is explicitly given by

$$\boldsymbol{W}_{\widehat{\boldsymbol{\theta}}_\ell}\boldsymbol{z}_{\ell-1}+\boldsymbol{b}_\ell=\begin{bmatrix}\boldsymbol{C}^{1,1}_{\widehat{\boldsymbol{\theta}}_\ell} & \cdots & \boldsymbol{C}^{1,C}_{\widehat{\boldsymbol{\theta}}_\ell}\\ \vdots & \ddots & \vdots\\ \boldsymbol{C}^{K,1}_{\widehat{\boldsymbol{\theta}}_\ell} & \cdots & \boldsymbol{C}^{K,C}_{\widehat{\boldsymbol{\theta}}_\ell}\end{bmatrix}\boldsymbol{z}_{\ell-1}+\begin{bmatrix}b^1_{\widehat{\boldsymbol{\theta}}_\ell}\\ \vdots\\ b^K_{\widehat{\boldsymbol{\theta}}_\ell}\end{bmatrix}. \tag{34}$$

We can of course use a series, cascade or stack of convolutional layers, and create what is known as a convolutional neural network (CNN) [51], [52], as shown, for example, in Fig. 6. Fig. 6 gives an example CNN trained to map low-dose PET images to higher dose equivalents. Yet, as should be clear from Figs. 5 and 6, CNNs can in fact have wide-ranging uses, even such as mapping ML-EM reconstructions to MAP-EM reconstructions, for accelerated reconstruction (Rigie *et al.* [53]). Mappings based on (34), with only convolutional layers, are known as fully convolutional networks (FCNs). However, convolutional and FC layers can be used together in a general CNN.

A final important note for this section is the universal approximation theorem (UAT) [54]. In general, for a deep neural network we have

$$\boldsymbol{z}_\ell=\sigma_\ell\left(\boldsymbol{X}_{\widehat{\boldsymbol{\theta}}_\ell}\boldsymbol{z}_{\ell-1}+\boldsymbol{b}_l\right) \tag{35}$$

where the matrix $\boldsymbol{X}$ can be any matrix, whether representing multiple multichannel convolutions, or a fully general linear mapping. It has been shown that exploiting the nonlinearity between layers allows many practical and useful mappings to be approximated well, if sufficient layers are used. This is a very important result, meaning that essentially any useful image processing mapping can in theory be replaced by a sufficiently well-trained deep network, offering complete flexibility in terms of inputs and desired outputs.

G. Encoders, Decoders, Generative Models, and GANs

Deep learning can be used for *representation learning* (or feature learning) whereby a network learns how to represent input information in a different way, which is useful to a desired task. Viewed this way, a deep network is just a change of representation of the input information, either lossless, or indeed discarding information irrelevant to the desired task of the mapping. Image reconstruction itself can be regarded as a change of representation of the very same information contained in the data—just represented in the form of an image instead of measured data. In this context, there are three important classes of deep networks that are explicitly identified as *encoders*, *decoders*, and *generators*. Strictly speaking any arbitrary layer or series of layers of a deep network can be regarded as any of these three classes, as it depends on the task and our interpretation of the representation at a particular stage or layer in a network, in terms of how we interpret the kinds of vectors (feature maps) going into, and out of, one or more layers.

An encoder transforms an input vector to a different representation or feature vector, often referred to as a *latent space*, which might be a more compact form, or a more useful representation. So we have, for example

$$\boldsymbol{z}=E\left(\boldsymbol{m};\widehat{\boldsymbol{\theta}_E}\right) \tag{36}$$

where E is the encoding operator learned from training data, which in general is a nonlinear operator specified by possibly many layers of encoder parameters, $\widehat{\boldsymbol{\theta}_E}$, corresponding to a cascade of mappings, each of which is given by the form of (35).

Preferably the encoded latent representation should not lose any information of interest, but should provide a representation which is more useful, such as being more compressed (lower dimensionality), semantically rich or in a form which simplifies the desired task which is to be performed on the input. In the context of PET imaging, an input vector may be

sinogram data or already directly interpretable as an image. A trivial untrained linear example of an encoder would be use of a number of rows of the DFT analysis matrix, often called the Fourier encoding matrix in the context of MR imaging. This encodes (analyzes) an input image into coefficients in k-space, for a set of sinusoidal functions of various spatial frequencies (k). If we were only interested in images of low spatial resolution, then a compact, limited (sparse) number of nonzero k-space coefficients can of course encode spatially extensive images with densely populated nonzero pixel values. More generally however, an encoder mapping is learned from training data rather than mathematically designed, and is also nonlinear. The nonlinearity of the encoder can be considered as either activating or eliminating various encoding analysis operators, in a fashion that depends on the particular input data. This can also be seen as partitioning the input domain, and having conventional linear encoders for different regions of the input domain [56].

A decoder transforms from a (coded or latent) representation into something that we might choose to identify directly as having explicit meaning or utility (so no longer coded), such as an image, although again, this is somewhat arbitrary, depending on our interpretation. A general decoder could be denoted

$$\widehat{\boldsymbol{x}} = D\left(\boldsymbol{z}; \widehat{\boldsymbol{\theta}_D}\right) \tag{37}$$

where D is the overall, generally nonlinear decoding operator, specified by the possibly many layers of decoder parameters, $\widehat{\boldsymbol{\theta}_D}$ (which would in general have been learned from training data). Similar to the encoder, this decoding operator could be given by a cascade of mappings, each of which is given by the form of (35).

A trivial untrained linear example would be the inverse DFT, which decodes a spectrum by using the k-space coefficients as the amplitudes of sine and cosine basis functions to synthesize an output. With the simple linear example of the DFT encoder and inverse DFT decoder, we could again consider using a compressed latent space representation of an image or signal, by only retaining or using a subset of the k-space coefficients. Using a random subset of k-space would correspond to one example of compressed sensing MR imaging, as a case of sparse coding. More generally though, the decoder mapping is nonlinear, and so can be broadly considered as using a set of learned representation basis vectors which are chosen according to the input code vector.

In the context of PET image reconstruction, dictionary learning is another good example of a decoder mapping. The goal in dictionary learning methods is to learn an image representation set of basis functions (learnable either from the data in hand, prior data, and/or data from another modality), and express the reconstructed image as a weighted sum (a synthesis) of those learned basis functions. The latent space is the set of coefficients for the basis functions, and these should either be non-negative or sparse, to ensure that only key signal is retained and that data noise cannot survive the representation. An example is the work of Tahaei and Reader [57].

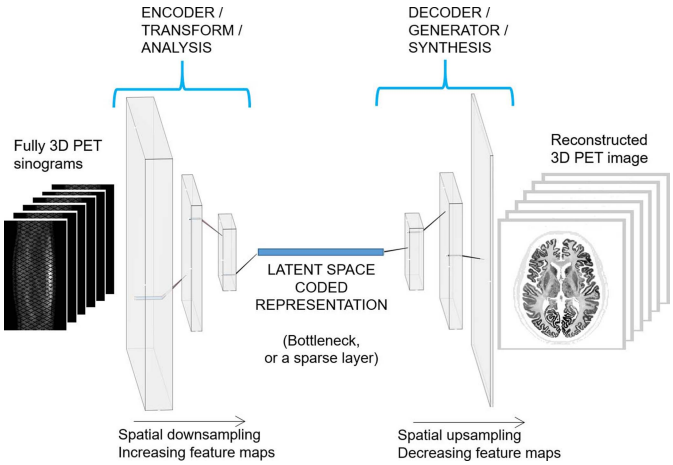

Fig. 7. Principles of an encoder (transform/analysis/compressing) operator and a decoding (generator/synthesis/decompression) operator, in this example case for mapping from fully 3-D sinograms, to a latent space, and then out to a 3-D PET image. This would correspond to the case of a direct mapping for PET reconstruction, to be covered later in this review. It is important to note that a decoder or generator can be used in isolation as an image generator for any given input vector (latent space or code vector).

Fig. 7 illustrates the principle: we seek a different, but useful, latent space, encoded representation of an image or sinogram data. This is such that, for example due to its compactness (reduced dimensionality or sparseness), in this latent representation space noise cannot be represented, but only information useful to the imaging task (e.g., clean and noise-free PET images). Also, tasks can be accomplished in simpler way in this latent space, and/or manipulations of data made easier (just as, by analogy, analysis and manipulations are sometimes easier in the Fourier domain than the space or time domain). With a coded description found in this clean latent representation space, we can then use a generator or decoder network to produce an end-point image corresponding to that representation. Of course, since the latent space should be designed to encode only the desired image features, the generated image will be composed only of such features.

A decoder can however be used in a more general sense, as we could freely design or randomly create our own latent space vectors, input these to a decoder, and so generate random sets of meaningful signals or images. Hence, the decoder can be used as a standalone *generator*, or a *generative model*. For the DFT example, this corresponds to designing or randomly choosing values in k-space, then applying an inverse DFT to *generate* images containing those spatial frequencies. Of course, random choices of spatial frequencies would lead to quite random output images. However, deep mapping decoders trained on useful image sets allow far more powerful and meaningful representations of images (beyond the simplicity of the Fourier basis vectors), such that randomly coded inputs result in new, never seen before, image samples drawn from highly complex high-dimensional probability density functions. Another simple example would be the KEM method, with its use of (16)—random positive values for $\boldsymbol{z}$ could be used, generating many different images, but all constrained to be within the manifold of objects composed from the dictionary of basis functions in $\boldsymbol{B}$.

Just as designation of an encoder, latent space, and decoder are all open to our interpretation, so also the demarcations between the encoder, latent space, and decoder are open to our interpretation. Consequently, there are infinitely many encoder operators, latent spaces, and decoders available for any given set of images. This can be seen by considering the very simple case of linear encoders and decoders: there are infinitely many linearly independent basis vectors that could be chosen for the encoder/decoder matrices [e.g., whether learned from data via principal or independent component analysis (PCA, ICA), or just mathematically devised such as the DFT].

Any given latent space and generator pairing models a probability density function, in the J-dimensional output image space. By using many example images to train an encoder, latent space, and decoder such that the input matches the output (an autoencoder), we can find a latent space, or better still a probability density function in the latent space (as done by a variational autoencoder [58]), such that random sampling of latent space vectors will map, via the decoder, to produce the distribution in the desired subspace/manifold of expected object vectors $\boldsymbol{x}$.

We can also provide nonrandom input vectors to generators, such as images or sinograms, in which case the generator becomes what is known as a *conditional generator* (Fig. 8). For example, sinogram data or a provisional image can be supplied to the network, and a high quality sample predicted from that conditional information. (In such cases, it is helpful to regard the conditional input information as first being encoded to a latent space, from which the generator then generates an image). When input images are used, this means that even very simple denoising mappings can in effect be regarded as conditional generators—they map a fixed input image to a fixed point in the output manifold. In contrast, of course, a fully fledged trained generator should, with random inputs, be able to always output meaningful images, populating the entire manifold of useful images based on the training data.

Training of generative models can be enhanced, to deliver more realistic results (i.e., more closely resembling samples from the training data), through use of a *discriminator*. So-called generative adversarial networks (GANs) train a second network, a discriminator, to impose improved performance of the generator network [7]. Discriminators improve the performance of a generator by learning to discriminate between real samples from the training data (drawn from the real probability density function in the manifold of the space of $\boldsymbol{x}$), and samples from the generator. The output of the discriminator is used as a penalty in the training loss function for the generator: if the discriminator can recognize a generated sample as being a generated one (a fake sample), this penalizes the loss function for the generator, such that it has to train better to seek a lower loss. The generator and discriminator are trained in an alternating manner, to reach a point whereby the generator can produce samples for which the discriminator only has 50% success rate in correctly classifying a synthetic sample as real or as synthesized. GANs have been applied in the context of MR reconstruction [59], and very recently

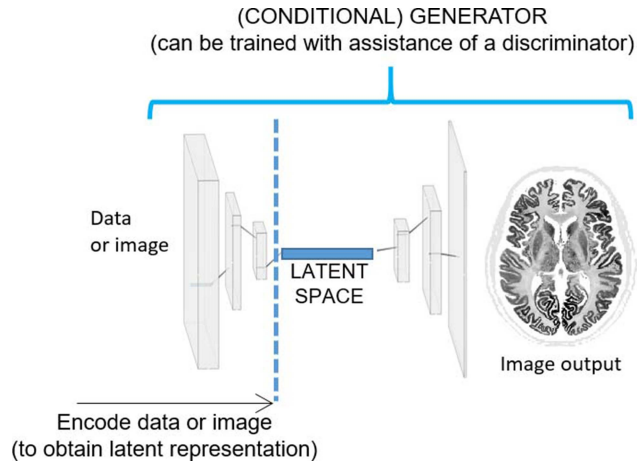

Fig. 8. Generators and conditional generators (GANs are a special case, where the generator has been trained with the assistance of a discriminator, encouraging outputs to look comparable to real data examples). Any network or mapping can in principle be regarded as a generator, in that an input will be mapped to an output, through a (*possibly only notional*) latent space. Hence, generators can even be regarded as including the encoder as well. Conditional GANs, when conditioned on specific input data or images, can be regarded as encoding the input, then generating an output, e.g., using a CNN or U-Net. As such, any image denoising operator can be viewed as a conditional generator (or conditional GAN if trained in conjunction with a discriminator), and are usually easier to implement than fully fledged generative models (which are required to generate meaningful output sample images when given random inputs, or random latent space values).

for PET reconstruction ([60], considered later in this review) as well as for post-reconstruction processing of PET images [55], [61], [62].

In summary, generators can be regarded as standalone decoders, or as synthesizers, in the form of a deep neural network which takes as direct input the latent space representation. The parameters of the generator are usually learned from unlabeled training data examples. These networks ideally should be able to generate all feasible reconstructed images of interest with appropriate probabilities based on the training samples.

We finish this section by mentioning the popular U-Net deep architecture [63]. This architecture very much follows the form of an encoder and decoder, but the critical difference is that there are skip connections included, which allow each downsampling section of the encoder mapping to be skipped, with feature maps at each downsampling stage being transferred directly as channel inputs to the respective upsampling (decoder) stage. Fig. 9 shows an example of this network. The use of skip connections allows higher resolution feature maps to be directly included for consideration as extra channel inputs for the decoder, and in effect serve to increase the expressive potential of the overall network. The approach has proven highly successful in the original application area of image segmentation, and PET image reconstruction methods have since also made use of U-Nets, which when supplied with input images can be regarded also as conditional generator networks. Nonetheless, improvements have subsequently been proposed, such as deep convolutional framelets, to overcome some of the limitations with U-Nets [64], [65].

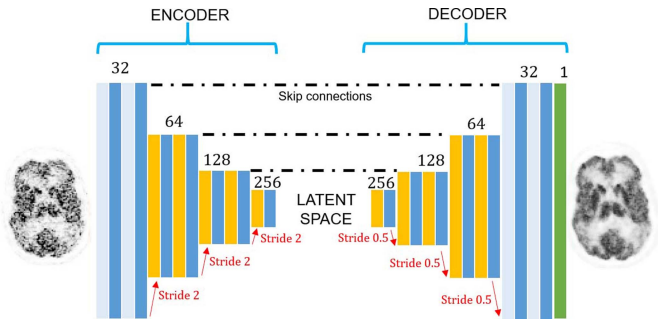

Fig. 9. Example of a U-Net architecture, in this case composed of a convolutional downsampling encoder (using stride 2 convolutions to downsample by a factor of 2) and a convolutional upsampling decoder (using fractional stride of 0.5 to upsample by a factor of 2). Crucially there are skip connections between each stage of down/up sampling, enabling greater levels of representation capacity, or expressivity, of the network.

H. Optimization and Generalization

From a conventional perspective, the highly parameterized nonlinear deep networks just described would be highly challenging to optimize. We finish this section by noting that the algorithmic technology, based on backpropagation, for seeking to minimize loss functions such as (18) has become available via toolboxes, such as TensorFlow and PyTorch, using gradient-based algorithms centered on stochastic gradient descent (SGD). We refer to excellent reviews of these optimization topics [66], regarding them as enabling technologies, permitting the design and practical training of deep nonlinear networks.

While it is already a significant endeavor to minimize a loss function with a highly nonlinear parameterization, there is also the further challenge of reducing the generalization error—i.e., how well the trained network performs when tested on new, never seen before test data. This is a major research area, with existing strategies including regularization of the loss function (e.g., norms of the parameters, to stop them becoming too large in uniquely fitting the training data only), dropout (i.e., randomly switching off a fraction of the parameters in a given FC or convolutional layer to stop them memorizing training data), data augmentation (i.e., artificially enlarging the domain of the training data by manipulating and processing existing training data), and transfer learning. Transfer learning concerns using networks previously trained for *other* tasks and data, and applying these to a new task. For PET this has been done in a post-reconstruction context, using a pretrained VGG network [67] to assist in PET denoising [68], and the VGG network has also been used in PET reconstruction, as will be mentioned below. Domain adaptation, similar to transfer learning, involves performing the same task but on different source domain data, such as for example data from different scanners [69] or from different PET centers. In the context of PET image reconstruction, generalization error reduction can be regarded as improving domain adaptation. The case of performing the same task on different domains has already been progressing at a rapid rate, for example, in the context of MR brain segmentation from different centers [70].

IV. Overview of Deep Learning in PET Reconstruction

There are at least four key ways in which to exploit the potential of deep learning mappings within PET image reconstruction, compared and summarized in Table I. One specific case which will not be covered in this article is that of post-reconstruction processing (e.g., [62] and [71]), as this no longer involves the reconstruction process. In this section we provide a brief overview of five key current approaches, before exploring four of them in greater detail in Sections V–VIII of the review.

A. Deep Learning for Direct Reconstruction

The first approach is a full end-to-end mapping, a direct reconstruction method, which uses a deep network to map from raw sinogram data $\boldsymbol{m}$ directly to an end-point reconstructed image estimate $\widehat{\boldsymbol{x}}$. This is represented simply by (as covered in Section III)

$$\widehat{\boldsymbol{x}} = F\big(\boldsymbol{m}; \widehat{\boldsymbol{\theta}}\big). \tag{38}$$

Hence, every aspect of the image reconstruction (the physics, imaging model, and statistics) needs to be learned by the deep mapping, which can require a large quantity of training data. In principle, these approaches avoid modeling errors, and once trained result in fast and potentially highly accurate reconstructions. Key examples will be covered in more detail in Section V below, but they tend to be characterized by relatively high training data needs, with high (even prohibitively so at present) computational demands for fully 3-D reconstruction.

B. Deep Learning for Image Generation: Synthesis Regularization

A second approach is to use deep learning only as an image constraint, requiring that an estimate of the image be represented by a deep mapping generator

$$\boldsymbol{x} = F(z; \boldsymbol{\theta}) \tag{39}$$

with all other components of the image reconstruction task corresponding to the conventional ones described earlier in Section II. The core idea of (39) is to require that any image estimate $\boldsymbol{x}$ be the output of a deep network F operating on some input code vector, z. A simple example is for the input code z to be a current noisy image, with the generator F only needing to be a denoiser, thereby regarding it as a conditional generator. Notably, this allows considerable flexibility for integrating sophisticated denoisers into PET image reconstruction, including fully 3-D reconstruction. These approaches will be covered in more detail in Section VI below.

C. Deep Learning for Analysis Regularization

A third approach is to use a deep network inside a conventional prior or penalty function, as just a component of an otherwise conventional image reconstruction method using an analysis regularization strategy. For example, any of the deep object models from a synthesis strategy (e.g., image generation/image synthesis/denoising) can be used, but instead of

TABLE I
SUMMARY OF CONVENTIONAL AND DEEP LEARNING-BASED PET RECONSTRUCTION, DISTINGUISHED ACCORDING TO THE WAY THE OBJECT (IMAGE), DATA MEAN, AND DATA NOISE ARE EACH MODELLED, AS WELL AS THE TYPE OF REGULARIZATION AND ALGORITHMIC PRACTICALITIES

<table>
<tr><th>METHOD</th><th>Model of object $f(\boldsymbol{r};\boldsymbol{x})$</th><th>Model of noise $D_{PET}(\boldsymbol{q}(\boldsymbol{x});\boldsymbol{m})$</th><th>Model of mean of data $\boldsymbol{q}(\boldsymbol{x})$</th><th>Regularisation $R(\boldsymbol{x})$</th><th>Algorithm</th><th>Training data needs</th><th>Generalisation capabilities</th><th>Memory needs</th><th>Execution speed</th></tr>
<tr><td>FBP
(conventional)</td><td>None</td><td>Gaussian</td><td>Radon / x-ray transform</td><td rowspan="2">None</td><td>One step</td><td rowspan="2">N/A</td><td rowspan="2">No training history, acquisition data only</td><td rowspan="3">Low (suited to fully 3D)</td><td>Fast</td></tr>
<tr><td>ML-EM
(conventional)</td><td rowspan="2">Voxels</td><td rowspan="5">Poisson</td><td rowspan="5">Factorised system model</td><td rowspan="5">Iterative</td><td rowspan="5">Slow</td></tr>
<tr><td>MAP-EM
(conventional)</td><td>E.g. Quadratic, guided, TV, Huber</td></tr>
<tr><td>Synthesis
(deep learned)</td><td><u>Learned</u></td><td>None</td><td rowspan="3">Low ~10-100 datasets</td><td rowspan="3">Training data, with updates from acquisition data</td></tr>
<tr><td>Analysis
(deep learned)</td><td rowspan="2">Voxels</td><td>Penalty for difference from <u>learned</u> object model</td><td rowspan="2">Variable (suited to fully 3D)</td></tr>
<tr><td>Unrolled
(deep learned)</td><td><u>Learned</u> penalty function</td></tr>
<tr><td>Direct
(deep learned)</td><td colspan="5"><u>Learned</u></td><td>One step</td><td>High ~1000-100000 datasets</td><td>Limited by training data</td><td>High (less suited to fully 3D)</td><td>Fast</td></tr>
</table>

imposing these as hard constraints, the analysis prior stipulates that a reconstructed image $\boldsymbol{x}$, while being optimized to agree with the measured data $\boldsymbol{m}$ (e.g., through the Poisson log likelihood), should not deviate too far from a deep denoised version of the image. This is less constraining than the synthesis approach [72], just as MAP-EM methods are, for example, less constraining than KEM methods (see previous Sections II-B and II-C).

D. Deep Learning for the Entire Prior: Unfolded Methods

A fourth approach is to use deep learning for the entirety of the penalty or prior, thereby completely discarding any analytic, intuitive, or handcrafted component. This means there is no chosen potential function and no explicit analysis operator, but instead the entire prior, including any effective potential function, is deep learned. To achieve this, iterative reconstruction algorithms can be *unrolled*, or *unfolded* into a series of modules or blocks, such that each and every iterative update is explicitly an update operator in a long cascaded series of blocks (using the gradient of the data fidelity term and the gradient of the penalty term). This long chain of processing blocks gives a deep overall mapping network, for which deep learning can be used to find the mapping which corresponds to where the gradient of the penalty is required. The overall network combines trainable components, (the gradient of the unknown penalty) and fixed operator components—i.e., the data-consistency update, usually derived from the gradient of the Poisson log likelihood for PET reconstruction. This approach can be viewed simply as interleaving partial or complete reconstruction operators with deep denoising operators, in repeated blocks. Each such block performs a number of MAP-EM image reconstruction updates (from just one, up to possibly even a completely converged reconstruction), using an analysis regularization based on a prior image. The prior image is a deep-learned denoised version of the previous reconstruction estimate. A deep network, which usually depends on the overall block number, is used to denoise the outcome of the reconstruction operator, in order to provide an updated denoised prior image for the next block. These unfolded networks will be considered in detail in Section VII below, and notably are generally practical for fully 3-D reconstruction.

E. Deep Learning for Preprocessing and Post-Processing

As mentioned, deep learning for post-reconstruction processing (or even for preprocessing of the raw sinogram data), is not under consideration in this review. In both cases, the approach is typically to upgrade low-dose PET data or images to high-dose equivalents, lowering noise and enhancing spatial resolution. There have now been numerous methods for post-reconstruction deep learned denoising in PET (e.g., [62], [71], [73], and [74]).

A noteworthy exception, which can be viewed as reconstruction (or possibly post-processing) is the use of backprojected images, whereby the raw PET data (sinograms or list-mode data) are first backprojected into a 3-D image array prior to application of a reconstruction algorithm to recover the quantitative radiotracer distribution. Exploiting backprojected images dates back a long time in PET (e.g., [44] and [75]), and more recently the backprojection can also exploit time-of-flight (TOF) information, to produce histo-images (distinct from the original proposal of [76] which has a histo-image for

TABLE II
DIRECT DEEP LEARNING METHODS FOR RECONSTRUCTION

Name	Architecture [total parameters]	Loss function / optimiser (max epochs)	Training data inputs and targets	Number of training dataset pairs
AUTOMAP Zhu *et al.* [12]	2 FC layers (with tanh) and CNN (3 or 4 layers, 2 with ReLU) [~800 million parameters]	MSE with L1 penalty / RMSprop (100)	Input: undersampled *k*-space data (128×128) Target: Fully sampled T1w MRI images (128×128)	50,000
DeepPET Häggström *et al.* [14]	Convolutional encoder-decoder: total of 31 C layers (with batch normalisation (BN) and ReLU between all C layers) [>60 million parameters]	MSE / Adam (150)	Input: 2D noisy sinograms (269×288) Target: Ground truth PET images (128×128)	203,305
DPIR-Net Hu *et al.* [60]	Convolutional encoder-decoder: total of 35 C layers (with BN and ReLU) [>60 million parameters] *Discriminator*: 8 C layers with BN and ReLU, 1 FC with LReLU and 1 FC with no ReLU	Wasserstein GAN + VGG* + MSE / Adam (100)	Input: 2D noisy sinograms (269×288) Target: Ground truth PET images (128×128)	37,872
DirectPET Whiteley *et al.* [79]	Encoding Segment: 3 C layers Radon Inversion Layer Segment: 28 FC layers + Refinement and Scaling segment: residual CNN with 23 C layers of 64 kernels [~385 million]	VGG* + MAE + MS-SSIM / Adam (1000)	Input: 16 2D sinograms normal or at 50% count level (400×168×16) Target: 16 2D image slices, normal reconstructions of 100% counts (400×400×16)	2,048
	FC layer: fully-connected layer C layer: convolutional layer	*VGG: perceptual loss based on VGG network [67]		

each view). Such TOF-backprojected images are excellent candidates for deep-learned mappings such as CNNs, as recently demonstrated with promising results [77].

V. DIRECT DEEP LEARNING PET IMAGE RECONSTRUCTION METHODS

In this section direct deep learning methods are considered in more detail, with Table II summarizing a comparison of key contributions. In particular, there are two pioneering examples of direct methods for PET image reconstruction, which have however only been applied to small 2-D slices (128×128): the methods of automated transform by manifold approximation (AUTOMAP) [12] and DeepPET [14]. Subsequent examples of direct reconstruction include Liu *et al.* [78] and Whiteley and Gregor [79] (which notably included multislice reconstruction), and more recently a version of DeepPET with a discriminator added on [60].

A. Direct: Fully Connected Layers With CNNs

Section III introduced FC layers and CNNs, and it should be clear that we are at liberty to combine these mappings sequentially, making a deeper network composed of both FC and convolutional layers. For example, a FC layer could be used to learn a mapping comparable to the inverse of the Radon transform [see previous (20) and discussion], and then a series of convolutional layers (a CNN) can be applied to the output of the FC layer in order to denoise via use of a more compressed representation. This is the approach of the direct deep learning method proposed by Zhu *et al.* in 2018, called AUTOMAP [12], shown in Fig. 10. AUTOMAP was proposed mainly for MR image reconstruction, but was also demonstrated for PET reconstruction. It has inspired other researchers to develop comparable methodology for PET (e.g., [79]).

The AUTOMAP architecture first reformats the complex MRI *k*-space data into a vector of real numbers only (for PET, this stage can be considered as just reshaping a PET sinogram into a column vector), followed by two FC layers (each with a tanh activation) to learn a mapping comparable to the inverse DFT in the case of MR, or comparable to an inverse of the Radon transform in the case of PET. This is followed by reshaping back to a 2-D image, ready for input to the CNN. The CNN in AUTOMAP is used to denoise by seeking to represent the image as a sparse collection of features found from the convolutional layers. Sparse features can be learned by the use of ReLU activation layers within the CNN used by AUTOMAP, (rather than by a bottleneck). This imposes a limited latent space for a compressed representation, occupying a limited manifold of the J-dimensional object space, mainly modeling real object features rather than noise features.

AUTOMAP reported good results for variously undersampled MRI reconstructions, although the PET reconstruction results were less convincing, with visual quality inferior to ordinary Poisson OSEM [80]. This was likely due to the use of single slice rebinned [81] input sinograms, precorrected for attenuation as input to AUTOMAP, and the fact that AUTOMAP had been trained with MR images which had undergone only a simple 2-D Radon transform followed by the

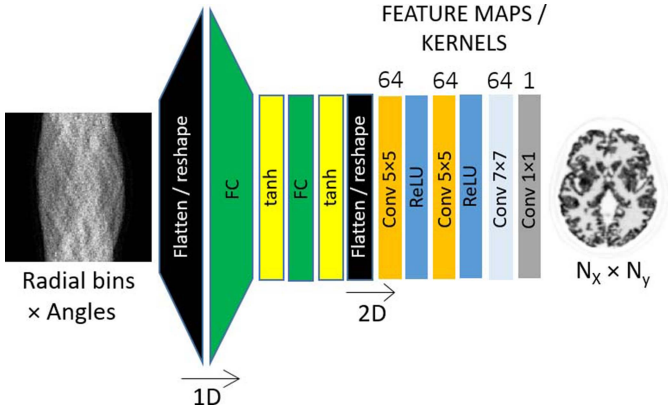

Fig. 10. Schematic of the AUTOMAP architecture [12], which starts with two main FC layers (each with tanh activation) followed by a CNN with three convolutional layers. This architecture was designed for MRI reconstruction, but was also demonstrated for PET reconstruction from sinogram data. The brain phantom used in this example (and in Figs. 11 and 14) is based on BrainWeb [105].

introduction of Poisson noise. Hence, the learned object manifold was for T1w MR images rather than [^{18}F]FDG-PET, and there was also a mismatch in the imaging model between the simulated training data and the test real data. Both of these limitations would have compromised the potential performance of the network.

B. Direct: Convolutional Encoder–Decoder

The principles of an encoder, latent space, and decoder have been applied to direct PET image reconstruction by Häggström *et al.* with their DeepPET architecture [14], as shown in Fig. 11. This convolutional encoder decoder (CED) was the second main proposed direct architecture for direct PET image reconstruction. Instead of using FC layers to map from a sinogram to the object space, the CED approach uses convolutional downsampling to transform progressively from the sinogram domain toward a learned latent space representation which has only very limited spatial sampling but is instead extremely rich in the number of features (latent variables). This latent space representation is then upsampled progressively in the decoder part of the network, in order to express the latent space information in the form of a PET image.

The input sinograms are precorrected, and so the network needs to learn a non-Poisson noise distribution. Häggström *et al.* report the benefits as greatly accelerated image reconstruction (up to 100 times faster), *de novo* learning of the imaging physics and the data noise distribution, thereby obviating any modeling assumptions in either regard.

Whilst the results reported are of high quality for the simulated data case, as would be expected due to a match in the imaging model used for training data and that used for the supplied test simulated data, the real data results (particularly for the brain data) still leave room for improvement. There is a need for high quality (ideally ground truth) reference data to go hand in hand with the measured data, in order to train the network correctly for real PET data.

An adversarial version of this kind of network was proposed by Liu *et al.* [78], with the key differences being the use of a U-Net (as a conditional generator) instead of the CED, and the addition of an adversarial/discriminator network. Subsequently, an extended version of the CED DeepPET with a discriminator added on was also proposed by Hu *et al.* [60].

VI. Deep Learning for Regularization Within Conventional Reconstruction

Recall that conventional model-based reconstruction, covered in Section II, used regularization via analysis or synthesis, but that one of the drawbacks was the use of handcrafted or mathematically convenient analysis or synthesis methods. Upgrading from a handcrafted prior to a data-driven one is a simple route for deep learning to bring benefits into conventional image reconstruction. The approaches reviewed in this section retain all the standard model-based reconstruction components (i.e., our knowledge of the imaging physics and statistics), but just use deep learning for where we are less certain and are in need of data/evidence-based prior information—the regularization component.

Generators can be used in an otherwise conventional image reconstruction framework, either in a synthesis capacity (whereby only images which are outputs of a generator can be used to optimize the reconstruction objective function) or in an analysis capacity (whereby transformation of the image by a network into a latent space should result only in a sparsely coded description).

A. Regularization by Deep-Learned Synthesis/Generative Models

There are three main approaches to a deep-learned synthesis model. The first is to estimate an input code vector $\boldsymbol{z}$ for a fixed deep network F [82], to deliver an image such that a reconstruction objective function (e.g., Poisson likelihood) is optimized. The second is to use a potentially arbitrary input code $\boldsymbol{z}$, whether random noise or a prior image, and estimate the network parameters $\boldsymbol{\theta}$ [13], in order to optimize the reconstruction objective function. Third, one could seek to estimate both $\boldsymbol{\theta}$ and also $\boldsymbol{z}$ in a simultaneous or alternating manner. A simple example is for the input code $\boldsymbol{z}$ to be a current noisy image, with the generator F only needing to be a denoiser, thereby regarding it as a conditional generator.

We will start with the case of estimating a code vector $\boldsymbol{z}$ directly, which when mapped through an operator produces the image vector $\boldsymbol{x}$. Recall from the introduction that the kernel method [KEM, (16) and (17)] was a synthesis method of regularization. This synthesis approach was in fact the motivation behind the work of Gong *et al.* [82], who, instead of using a linear set of basis functions $\mathbf{B}$, used a pretrained CNN as the generative mapping (fixed choice of $\boldsymbol{\theta}$ for a fixed mapping F), imposing the following model for the radiotracer distribution's parameter vector $\boldsymbol{x}$ [see again (1)]:

$$\boldsymbol{x} = F(\boldsymbol{z}; \boldsymbol{\theta}_{\text{FIX}}) \tag{40}$$

where the goal is to estimate the representation parameter code vector $\boldsymbol{z}$ such that when it is mapped through the fixed generator CNN F, an image $\boldsymbol{x}$ is delivered which is consistent with the data. However, of course, the constraints of the CNN representation will mean that the forward model of $\boldsymbol{x}$ will not be

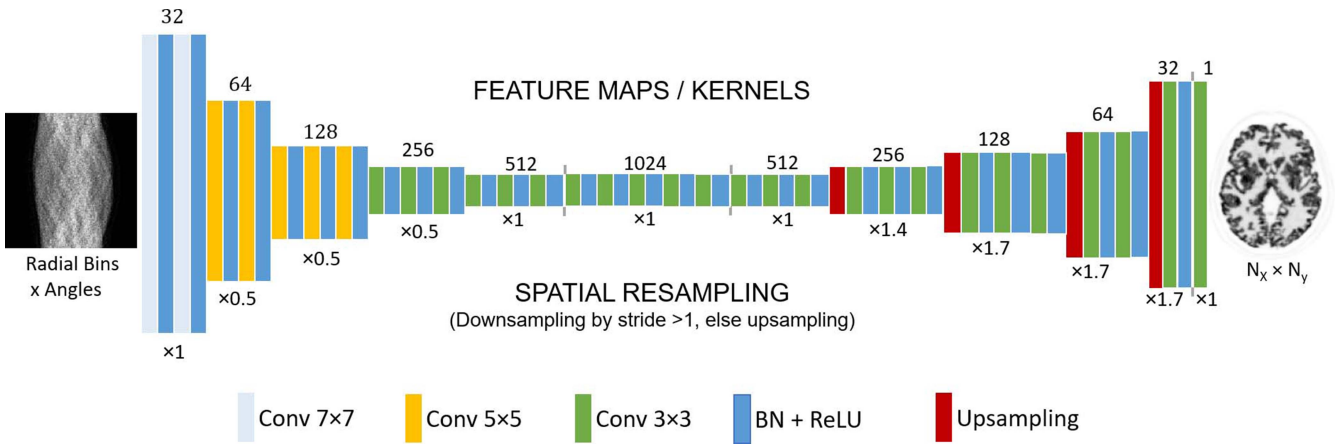

Fig. 11. Schematic of the CED architecture used by DeepPET for direct PET image reconstruction [14] and extended by inclusion of a discriminator by DPIR-Net [60]. PET scan information expressed in the sinogram domain is progressively transformed by simultaneous reduction of spatial sampling and increasing of the number of feature maps, until a feature-rich latent space representation is obtained. This latent representation is decoded back out to an image-space representation of the same information, by increasing the spatial sampling and reducing the number of feature maps.

entirely consistent with the data, due to both the CNN's constraints and the fact that the data contains noise. The estimation of z is purely by use of a constrained maximum-likelihood objective function, using (5) (the Poisson log likelihood)

$$\widehat{\boldsymbol{x}} = \underset{\boldsymbol{x}}{\operatorname{argmax}}\, L(\boldsymbol{x} \mid \boldsymbol{m}) \quad \text{s.t. } \boldsymbol{x} = F(z; \boldsymbol{\theta}_{\text{FIX}}) \tag{41}$$

where L is the Poisson log likelihood, given by the negative of (5). Given the constraint in (41), that a nonlinear CNN mapping must generate the solution vector $\boldsymbol{x}$, we can no longer use the conventional EM algorithm which assumes a linear forward model operating on the representation parameter vector. Using the approach of an augmented Lagrangian, Gong *et al.* first convert the constrained maximization problem of (41) into a penalized unconstrained problem instead. This results in integrating conventional reconstruction methodology into the broader algorithmic framework of the alternating direction method of multipliers (ADMM).

The first step of the method is a conventional MAP-EM problem to find an update of the image $\boldsymbol{x}$, using a quadratic penalty with a prior image [see (14)] $F(z^n)$ obtained by the CNN operating on the current estimate of the code vector z^n

$$\boldsymbol{x}^{n+1} = \underset{\boldsymbol{x}}{\operatorname{argmax}}\, L(\boldsymbol{x}|\boldsymbol{m}) - \frac{\rho}{2}\left\|\boldsymbol{x} - \left(F\left(z^n; \boldsymbol{\theta}_{\text{FIX}}\right) - \boldsymbol{\mu}^n\right)\right\|^2 \tag{42}$$

where $\boldsymbol{\mu}$ is initially zero, and r relates to the strength of the penalty. Equation (42) is solved by a conventional MAP-EM algorithm such as (15) given earlier. The latent code z^n is subsequently updated to seek to match this new image estimate

$$z^{n+1} = \underset{z}{\operatorname{argmin}}\, \frac{\rho}{2}\left\|F(z; \boldsymbol{\theta}_{\text{FIX}}) - \left(\boldsymbol{x}^{n+1} + \boldsymbol{\mu}^n\right)\right\|^2. \tag{43}$$

Equation (43) is solved by nonlinear least squares [the authors used a first-order approximation of the gradient of the objective function (44) with respect to z]. Finally, $\boldsymbol{\mu}^n$ is updated by

$$\boldsymbol{\mu}^{n+1} = \boldsymbol{\mu}^n + \boldsymbol{x}^{n+1} - F\left(z^{n+1}; \boldsymbol{\theta}_{\text{FIX}}\right) \tag{44}$$

where in effect $\boldsymbol{\mu}$ is an image showing the data-unique features which had not been expressed by the CNN, as it corresponds to the discrepancy between the new reconstructed image estimate $\boldsymbol{x}^{n+1}$ (based on agreement with the data) and the constrained output of the CNN operating on the code vector $F(z^{n+1})$.

The ADMM approach then reverts back to (42) to repeat the series of three updates. It can be seen that in effect, after the first iteration, $\boldsymbol{\mu}$ increases the penalty in the MAP-EM image reconstruction stage to encourage the image $\boldsymbol{x}$ to agree more with the CNN output (which is good if it denoises, but bad if it loses true image features). In a similar manner, for the update of the latent code z, the effect of $\boldsymbol{\mu}$ is now to emphasize importance of the data-unique features which had not been successfully represented in the previous iteration. It requires the network output to agree more with the data-based reconstruction, emphasizing regions of the image where there had been disagreements. Where the data contains features which are not readily expressed by the network F, extra penalties occur, to seek to reduce the discrepancy of network output with the data.

Gong *et al.* report improved lesion contrast, for a given noise level, compared to post-reconstruction CNN denoising (see Fig. 12), which perhaps is not surprising, as post-reconstruction CNN denoising no longer demands agreement with the original raw data, whereas the CNN representation method does.

The same authors extended their work, inspired by the "deep image prior" (DIP) [83] to use instead a *fixed* input vector z, defined to be, for example, the patient's MR image (the original DIP used random noise), and then trained a conditional generative CNN mapping F, such that the parameters of the network map the fixed z to the current reconstructed image [13]. The algorithm follows a similar framework to that just described in (42) to (44), with the key difference being that instead of updating the latent code vector z in (43), Gong *et al.* now update the network parameters instead

$$\boldsymbol{\theta}^{n+1} = \underset{\boldsymbol{\theta}}{\operatorname{argmin}}\, \frac{\rho}{2}\left\|F\left(z^{\text{FIX}}; \boldsymbol{\theta}\right) - \left(\boldsymbol{x}^{n+1} + \boldsymbol{\mu}^n\right)\right\|^2. \tag{45}$$

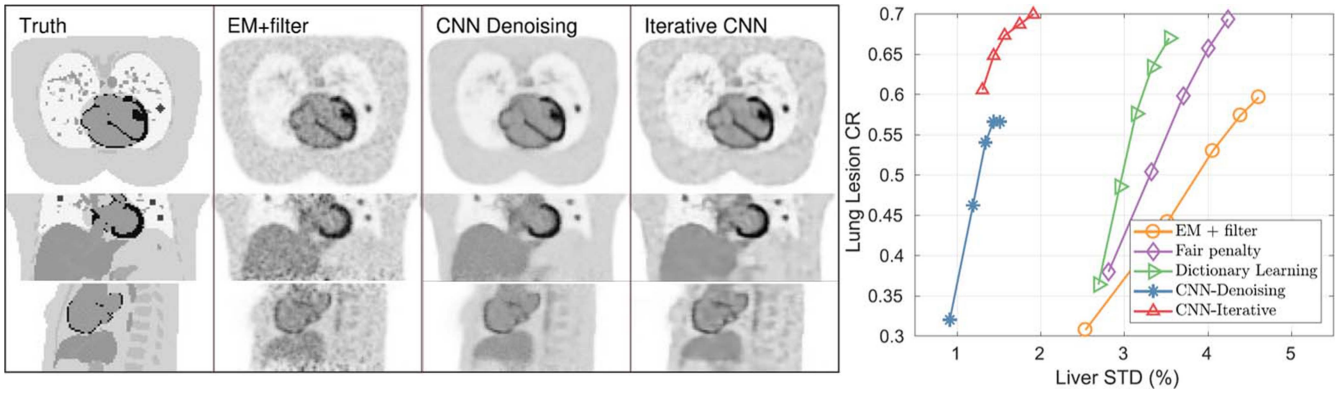

Fig. 12. Example results of using the CNN representation method ("Iterative CNN") compared to post-reconstruction CNN denoising (which does not involve consistency with the data) [82]. For a given standard deviation (STD) in the liver region, the contrast recovery (CR) in the lesion in the lung is better when using a CNN representation of the image. Both CNN methods outperform the 3 conventional methods, by lowering image noise in the liver region by up to a factor of 2.

Equation (45) is solved by training a deep CNN to map the fixed prior z^{FIX} (the subject's MR image) to match the current MAP-EM update x^{n+1}, with an emphasis (given by μ) in regions where the CNN had previously failed to represent features in the data-based reconstruction x. The crucial point to note is that the method does not use any training data, and can be viewed as unsupervised deep learning. Gong *et al.* compared the method to using a CNN penalty method, as shown in Fig. 13, where it can be seen that PET-unique regions are more clearly defined in the DIP method. The method was subsequently extended to 4-D image reconstruction for the Patlak model [84]. In the context of dynamic PET image reconstruction, Yokota *et al.* [85] used U-Nets as representations of parametric images, with random z inputs, thereby placing more demand on the network training. Furthermore, a GAN approach, which enhances the generator, has also been proposed [86].

Table III provides a representative summary of methods for synthesis-based deep learning methods in the literature. The limitations of these methods are that the hyperparameters, mainly r, need selecting for the components of the ADMM optimization. However, performance is intended to be independent of r, given the original objective function (41) is purely an unpenalized maximum likelihood with just an object model constraint.

B. Regularization by Deep-Learned Analysis

A more flexible (less constrained) approach to using deep generators in reconstruction is via a regularization analysis framework. This allows a balance between data-fidelity expressed at the pixel/voxel level and a penalty for deviation from a constrained, learned object model. An example would be a quadratic penalty

$$R(x) = \sum_{j=1}^{J} \left(\left[F\left(x^n; \theta_{\mathrm{FIX}}\right) \right]_j - x_j \right)^2 \tag{46}$$

where in (46) F would be a denoiser, or a conditional generator. Alternatively, the prior can stipulate that the reconstructed

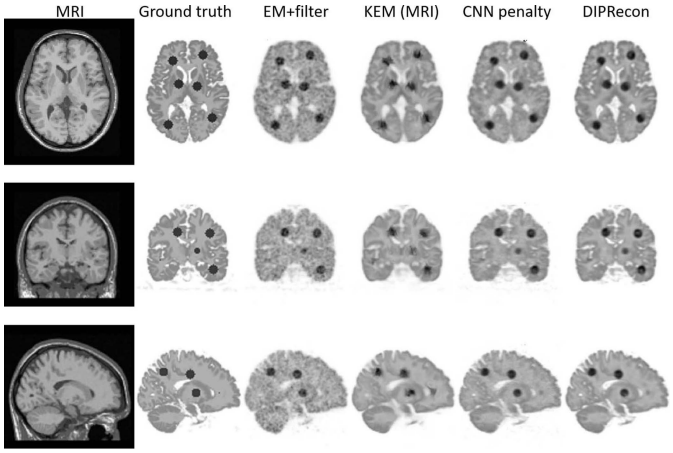

Fig. 13. Example results [13] for the deep image prior (DIPRecon), offering improved lesion contrast compared to the kernel method (KEM with MRI) and a CNN penalty method.

image should not deviate too far from a sparse-coded image

$$R(x) = \sum_{j=1}^{J} \left(\left[F\left(z^n; \theta_{\mathrm{FIX}}\right) \right]_j - x_j \right)^2 \tag{47}$$

where, depending on the architecture of the mapping, it may also be necessary to explicitly require sparsity of the code vector z, by adding a penalty for highly populated code vectors which deliver a large norm

$$R(x) = \sum_{j=1}^{J} \left(\left[F(z; \theta) \right]_j - x_j \right)^2 + \|z\|. \tag{48}$$

A penalty comparable to (48) was the approach used by Xie *et al.* [87]. In these formulations it is noted that a conventional potential function, such as a quadratic as shown in (14), is still used, as was the case in Kim *et al.* [88]. Otherwise, all other image reconstruction components remain conventional in definition. Gong *et al.* [13] also used a CNN penalty as an example method to compare with in their work using the deep image prior, as was shown in Fig. 13.

TABLE III
SYNTHESIS REGULARIZATION DEEP LEARNING METHODS FOR PET RECONSTRUCTION

Name	Architecture [total parameters]	Loss function / optimiser (max epochs)	Training data inputs and targets	Number of training dataset pairs
CNN Representation Gong *et al.* [82]	Modified U-Net: 3 encoding blocks (C+BN+ReLU), 3 decoding blocks (C+BN+ReLU), 2 C+BN+ReLU in bottleneck, skip connections (sum residual and current image) instead of concatenating (add residual as a new channel) [~1.4 million parameters]	MSE / Adam (1500)	Input: 3D low-count reconstructions Brain data**:** (128×128×91), Lung data: (128×128×49) Target: 3D high-count reconstructions Brain data**:** (128×128×91), Lung data: (128×128×49)	Brain: 15 Lung: 5
Deep Image Prior Gong *et al.* [13]	Modified U-Net: 3 encoding blocks (2 C+BN+LReLU), 3 decoding blocks (2 C+BN+LReLU), 2 C+BN+LReLU in bottleneck, skip connections instead of concatenating [~2 million parameters]	MSE / L-BFGS (20 epochs per ADMM iteration, 100 ADMM iterations in total)	Input: 3D anatomical prior (MRI) (192×192×128) Target: An EM iteration (3D) (192×192×128)	1
Xie *et al.* 2019 [86]	GAN: Generator: Modified U-Net: 3 encoding blocks (with 1 C layer), 3 decoding blocks (with 1 C layer) and a self-attention module and 2 C layers at bottleneck [~1.5 million parameters] *Discriminator*: 6 C layers with self-attention module in between 3rd and 4th layer	Generator: MSE + Binary Cross-Entropy Discriminator: Binary Cross Entropy / Optimiser not specified (400)	Input: 3D low-count image (128×128×5) Target: High-count middle slice (128×128×1) + Label Real/Fake image	20
Yokota *et al.* [85]	3 U-Nets combined in parallel [~4 million parameters for 1 U-Net]	MSE / Adam (20,000)	Input: noise Target: spatial factors i.e. homogeneous tissues with kinetic parameters (128×128×3)	1

VII. UNROLLING OR UNFOLDING ITERATIVE RECONSTRUCTION WITH DEEP LEARNING

The direct deep learning methods previously described in Section V do not make any use of the imaging system model **A**, nor the statistical-noise model described in Section I. Instead, large quantities of training data are needed to learn these from scratch (see Table II). This is potentially advantageous, as it avoids modeling errors, but arguably it is wasteful, discarding years of progress in modeling expertise and reconstruction algorithm development. Furthermore, by excluding these models, there is potential for the direct methods to perform potentially inexplicable mappings, which may limit confidence, especially for unexpected (out of domain) inputs to the network.

In contrast, the deep learning regularization methods covered in Section VI do make use of existing models, exploiting deep learning only for image regularization. However, these approaches still retain the mathematically convenient potential functions which operate on these images for the regularizing penalty. The choice of potential function is not motivated by the data, but only by convenience.

There has been increasing work in physics-informed AI/deep learning (e.g., [89]), and image reconstruction is no exception. The goal here is to combine the power of the AI paradigm with our existing knowledge of the imaging physics and statistical modeling, seeking a hybrid new image reconstruction methodology that exploits the best of AI with the best of our understanding of imaging physics and reconstruction. This has the further advantage of using AI only for the parts of the reconstruction process for which we are not confident—such as precisely how to regularize, and to what strength, leaving the imaging system physics model and noise model to be what we are confident and know they should be. This has the advantage of interpretability, important for clinical imaging such as PET. The methods of Section VI have taken steps toward this, but as mentioned, do retain a convenient handcrafted potential function. The methods in this section will now also replace the potential function via deep learned mappings, based on unrolling conventional iterative reconstruction.

Fig. 14 shows the general framework for three of the key methods which will be covered in detail below. The first proposal of turning an iterative reconstruction method into an unfolded deep network was in fact by Gregor and LeCun, as early as 2010 [90]. Examples for medical imaging reconstruction include, from the world of MRI, the work of Hammernik *et al.* [91] (named a variational network). However, this section will focus on PET image reconstruction.

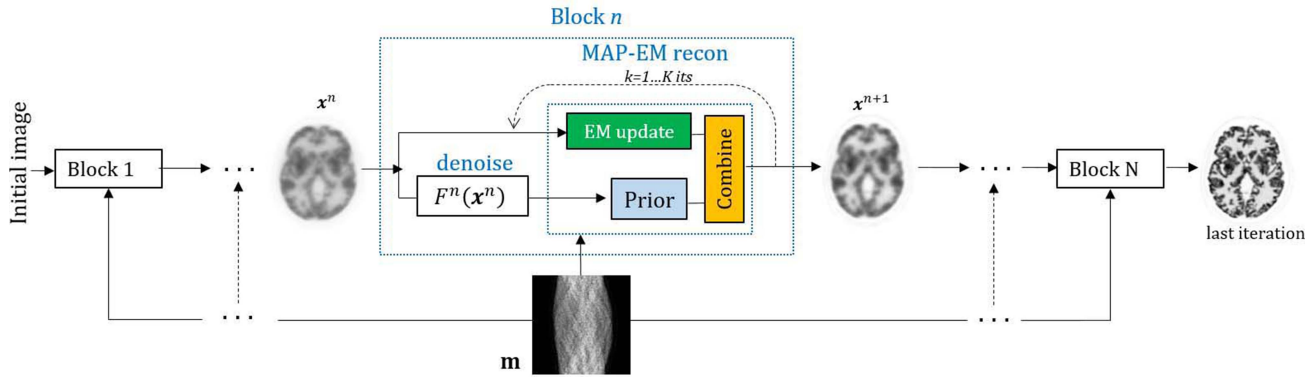

Fig. 14. General framework for three major unrolled methods for PET image reconstruction with integration of deep learning for the regularization. The unrolled series of updates is composed of $n = 1 \cdots N$ blocks. For BCD-Net, training is done at the block level, where the goal is to denoise an update to make it best match a high quality (or true) reference. In contrast, both MAPEM-Net and FBSEM-Net conduct their training based on the very end image (last iteration), necessitating backpropagation through all N blocks during training in order to update the parameters for the denoiser network. For MAPEM-Net, there are $K = 2$ MAP-EM updates within a block, and training of each block-dependent denoiser F^n (depends on n) is such that the very last iteration matches the high quality reference (such as the last iteration of ML-EM from high quality data). For BCD-Net, K can vary from 1 to many iterates, and training is done for each individual block's denoiser F^n, such that the iteration at that stage matches the high quality reference-this avoids backpropagation through the whole series of blocks. For FBSEM-Net, $K = 1$, and training is such that the last iteration should match the high quality reference (e.g., high quality ML-EM reconstruction, or MAP-EM with light regularization from higher count data). The "Prior" indicates a fixed image used in an L2 norm penalty for the MAP-EM update.

A. EM-Net and MAPEM-Net

We first consider the method of Gong *et al.*, named EM-Net [92], which writes a general MAP-EM iterative update as follows:

$$\boldsymbol{x}^{n+1} = \boldsymbol{x}^n - \alpha\left[\boldsymbol{A}^T\left(\boldsymbol{I} - \frac{\boldsymbol{m}}{\boldsymbol{A}x^n + \boldsymbol{\rho}}\right) + \beta R'\left(\boldsymbol{x}^n\right)\right] \tag{49}$$

where a is the update step size, the first term in the square parentheses is the negative of the gradient of the Poisson log-likelihood data-fidelity term, and R' is the gradient of R, evaluated at $\boldsymbol{x}^n$. EM-Net replaces R' by a deep network, to obtain

$$\boldsymbol{x}^{n+1} = \left\{\boldsymbol{x}^n - \alpha^n \frac{\boldsymbol{x}^n}{\boldsymbol{A}^T\boldsymbol{I}}\left[\boldsymbol{A}^T\left(\boldsymbol{I} - \frac{\boldsymbol{m}}{\boldsymbol{A}x^n + \boldsymbol{\rho}}\right) + F\left(\boldsymbol{x}^n; \boldsymbol{\theta}\right)\right]\right\}_+ \tag{50}$$

where the step size a is learned and iteration (n) dependent, and a non-negativity constraint is imposed. Just one same trained U-Net is used for F for all iterations, to learn the gradient of the unknown penalty function. The training of this single mapping F, and the step sizes, was based on a MSE loss function which required the last iterative output, $\boldsymbol{x}^N$, for noisy data, to match the last iterative output from a reconstruction from high count reference data. However, Gong *et al.* subsequently reported that direct replacement of the gradient of the prior by a CNN may be too smooth to capture its required high spatial frequency components, and hence they proposed MAPEM-Net [93], another unfolded method.

The approach of MAPEM-Net is to extend the constrained ML problem (41), as was used for a deep learned image generator/CNN representation) to be now a constrained MAP problem instead

$$\begin{aligned} \widehat{\boldsymbol{x}} &= \underset{\boldsymbol{x},z}{\operatorname{argmax}}\, L(\boldsymbol{x} \mid \boldsymbol{m}) - \beta R(z) \\ \text{s.t. } \boldsymbol{x} &= z. \end{aligned} \tag{51}$$

Following a similar ADMM algorithmic approach to that covered in the earlier section for synthesis deep learned regularization (42)–(44), the following updates are obtained:

$$\boldsymbol{x}^{n+1} = \underset{\boldsymbol{x}}{\operatorname{argmax}}\, L(\boldsymbol{x} \mid \boldsymbol{m}) - \frac{\rho}{2}\left\|\boldsymbol{x} - (z^n - \boldsymbol{\mu}^n)\right\|^2 \tag{52}$$

$$z^{n+1} = \underset{z}{\operatorname{argmin}}\, \frac{\rho}{2}\left\|z - \left(\boldsymbol{x}^{n+1} + \boldsymbol{\mu}^n\right)\right\|^2 + \beta R(z) \tag{53}$$

$$\boldsymbol{\mu}^{n+1} = \boldsymbol{\mu}^n + \boldsymbol{x}^{n+1} - z^{n+1}. \tag{54}$$

Updates (52) and (54) compare directly with updates (42) and (44), but with z replacing $F(z)$. The update for $\boldsymbol{x}$, as before in (42), is readily achieved by one or more iterations of MAP-EM. However, the key change is for the z update, (53), which now includes a penalty $R(z)$, and Gong's implementation opts for replacing the entirety of update (53) by an iteration-dependent deep network acting to denoise the MAP-EM output, to obtain

$$z^{n+1} = F^n\left(\boldsymbol{x}^{n+1}\right). \tag{55}$$

The resulting algorithm is therefore very simple. One or more MAP-EM updates are performed for (52) (Gong *et al.* chose just two updates), then the result is denoised via a deep network [Gong *et al.* used an iteration-dependent U-Net for (55)]. This denoised image is then used as the prior image for the quadratic penalty in the next set of one or more MAP-EM updates [for solving problem (52) again]. It is important to note that while (52) uses an L2 norm (quadratic) penalty, nonetheless the prior being used in the reconstruction, $R(z)$, is completely learned and so is very unlikely to correspond to a quadratic potential.

The approach is shown schematically later on in Fig. 14, in a framework enabling comparison to two rival methods.

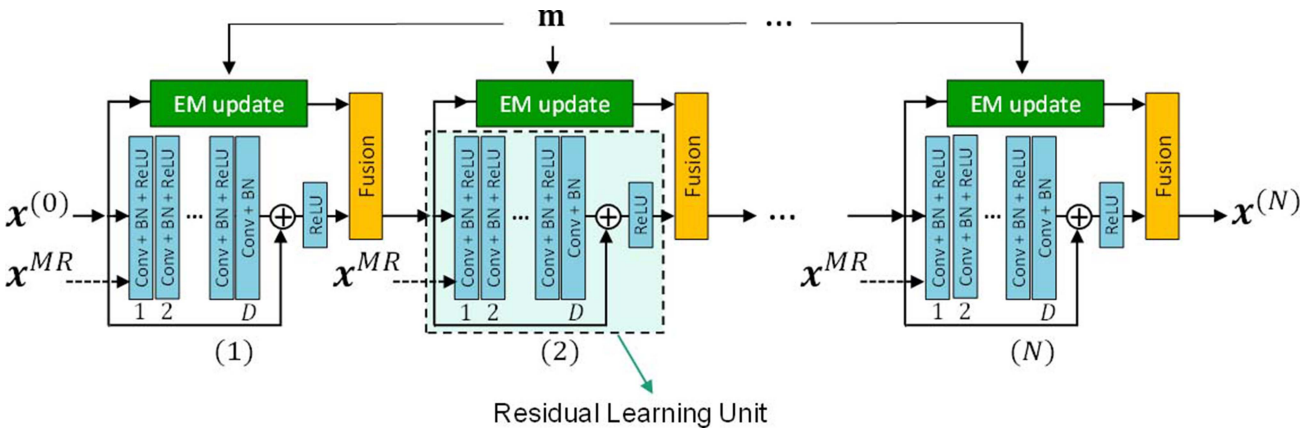

Fig. 15. Explicit schematic of the FBSEM-Net method [94], whereby a CNN with a skip connection, a "residual learning unit" is trained, along with the hyperparameter for the fusion of the denoised image with the EM update [(60) expresses the fusion step explicitly].

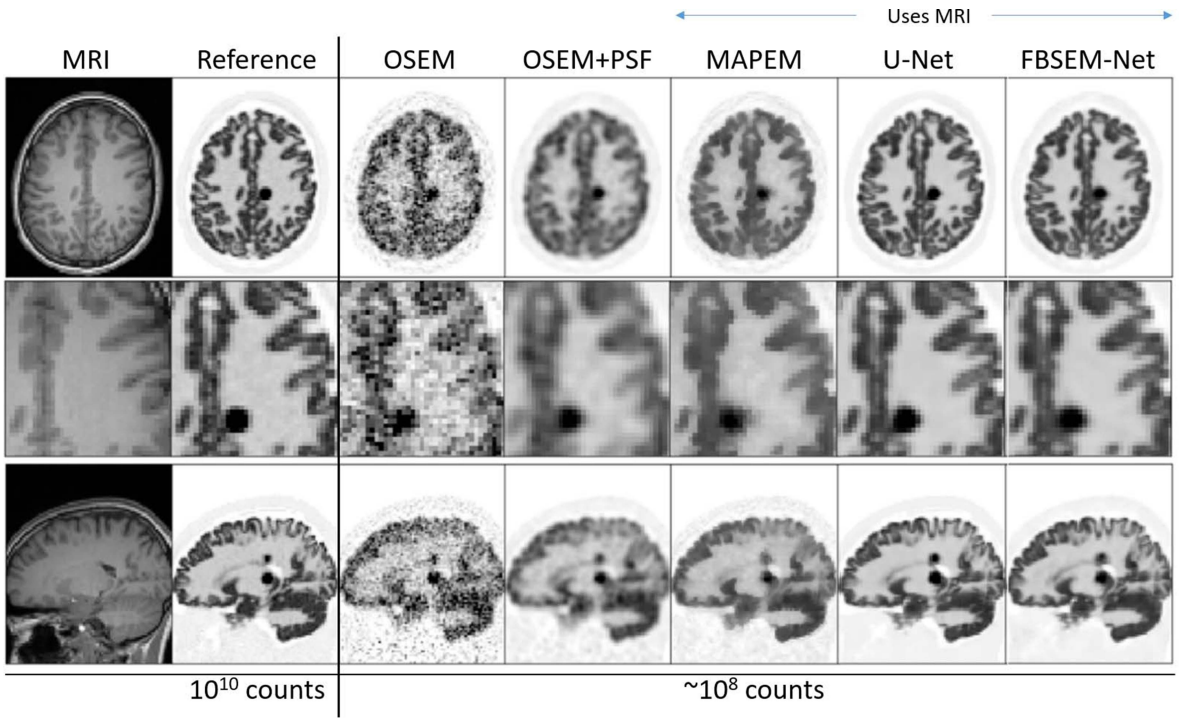

Fig. 16. Example slices for 3-D simulated [^{18}F]FDG data for FBSEM-Net, trained to match high-count reference data, when using ~100 times less data along with a T1w MR image for further information. FBSEM-Net is compared to conventional OSEM (no MRI benefit), without and with point spread function (PSF) modeling, MAP-EM with MRI guidance, and to a post-reconstruction denoised reconstruction using a U-Net supplied with MRI information.

B. FBSEM-Net

The unrolled method of Mehranian and Reader [94] is derived from the forward-backward splitting (FBS) algorithm [95] for solving the penalized Poisson log-likelihood. First, for a current estimate $\boldsymbol{x}^{n-1}$, the denoising (regularization) update is given by

$$\boldsymbol{x}^{n}_{\text{Reg}} = \boldsymbol{x}^{n-1} - \gamma\beta R'\left(\boldsymbol{x}^{n-1}\right) \tag{56}$$

which is a gradient descent toward the minimum of R with a step size of γ. The actual update of the current image estimate is then given by

$$\boldsymbol{x}^{n} = \underset{\boldsymbol{x}}{\text{argmax}}\, L(\boldsymbol{x}|\boldsymbol{m}) - \frac{1}{2\gamma}\left\|\boldsymbol{x} - \boldsymbol{x}^{n}_{\text{Reg}}\right\|^2 \tag{57}$$

which is a proximal mapping [just as used elsewhere, e.g., (52) for MAPEM-Net] associated with the Poisson log-likelihood L with γ as a regularization hyperparameter that limits the degree of data fidelity of $\boldsymbol{x}$ to $\boldsymbol{m}$ by requiring proximity to $\boldsymbol{x}^{n}_{\text{Reg}}$. Note the difference in (57) compared to (52) is the absence of the residual image (Lagrange multiplier) $\boldsymbol{m}$.

Following the approach of De Pierro [96], a separable surrogate is then defined for the objective function in (57), so that it can be rewritten as:

$$\boldsymbol{x}^{n} = \underset{\boldsymbol{x}}{\text{argmax}} \sum_{j} x^{n}_{j,EM}\ln\left(x_j\right) - x_j - \frac{1}{2\gamma s_j}\left(x_j - x^{n}_{j,\text{Reg}}\right)^2 \tag{58}$$

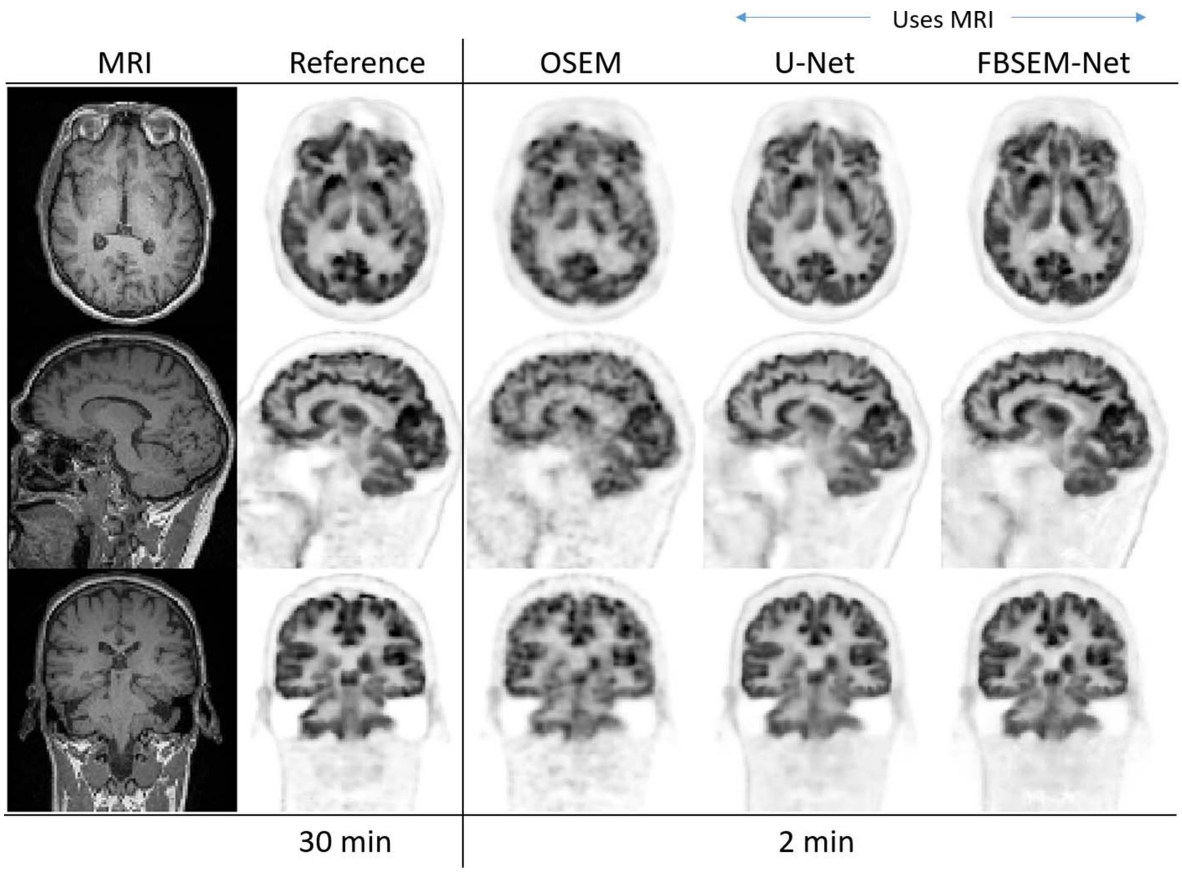

Fig. 17. Example results for real [^{18}F]FDG data for FBSEM-Net, trained to match 30 min reference data, when using 2 min data along with a T1w MR image for further information. FBSEM-Net is compared to conventional OSEM (no MRI benefit) and to a post-reconstruction denoised reconstruction using a U-Net supplied with MRI information.

where $\boldsymbol{x}^n_{EM}$ is given by the standard EM update [compare to the earlier (6) in Section II]

$$x^n_{j,EM} = \frac{x^{n-1}_j}{s_j} \sum_i \frac{a_{ij} m_i}{\sum_k a_{ik} x^{n-1}_k + \rho_i}, \; s_j = \sum_i a_{ij}. \quad (59)$$

By setting the derivative of the surrogate objective function (58) to zero, a closed-form solution is obtained [97]

$$x^{n+1}_j = \frac{2x^n_{j,EM}}{\left(1 - \delta_j x^n_{j,\text{Reg}}\right) + \sqrt{\left(1 - \delta_j x^n_{j,\text{Reg}}\right)^2 + 4\delta_j x^n_{j,EM}}}$$
$$\delta_j = \frac{1}{\gamma s_j} \quad (60)$$

which compares with the MAP-EM update (15) given back in Section II. For FBSEM-Net, the gradient of the prior in (56) is replaced by a CNN, and hence the whole update (56) becomes a residual network (i.e., using a skip connection)

$$\boldsymbol{x}^n_{\text{Reg}} = F\left(\boldsymbol{x}^{n-1}\right) \quad (61)$$

where the training of the deep network F, including the training of γ, is such that the end iteration after a series of updates matches a high quality image (e.g., a large number of iterations of ML-EM from high count data, or a lightly regularized MAP-EM if the count level is not sufficiently high in the reference data). Equation (61) compares closely to (55) in the method of MAPEM-Net, but here a fixed network F is used, not changing from iteration to iteration. These equations lead to a training framework which is also shown in Fig. 14, and more specifically in Fig. 15. Figs. 16 and 17 show example results from this approach for simulated and real data, respectively, including comparison with post-reconstruction deep denoising via a U-Net. The learned prior, which in the results shown also exploits an MR image as an input channel to assist the deep denoiser, notably enhances reconstructed image quality compared to conventional reconstruction. More details are available in [94].

C. BCD-Net

The method called BCD-Net [98] was adapted to low-statistics PET reconstruction by Lim *et al.* in 2018 [99], [100], predating EM-Net, MAPEM-Net, and FBSEM-Net. BCD-Net is very similar to the aforementioned methods, again interleaving a MAP-EM reconstruction (composed of just one or potentially very many iterations) with a deep learned denoising of the reconstruction update. The processed reconstruction is then used as a prior in the next full MAP-EM reconstruction. This iterative process continues, and the deep-learned processing of the reconstruction depends on the overall iteration number.

BCD-Net first conducts an initial number of EM iterations to get a current image estimate $\boldsymbol{x}^n$, which is denoised by a block

TABLE IV
UNROLLED PET RECONSTRUCTION WITH DEEP LEARNING: ARCHITECTURES

Name	Architecture [total parameters]	Loss function / optimiser (max epochs)	Training data inputs and targets	Number of training dataset pairs
EM-Net Gong *et al.* [92]	**1** modified U-Net shared for all blocks: 3 encoding stages (2 C+BN+LReLU), 3 decoding stages (2 C+BN+LReLU), 2 C+BN+LReLU at bottleneck, skip connections (add) instead of concatenating [~2 million parameters]	MSE / Adam	Input: last iteration after all blocks Target: 3D high count reconstruction (128×128×46)	16
MAPEM-Net Gong *et al.* [93]	**8** modified U-Nets, each composed of 3 encoding stages (2 C+BN+LReLU), 3 decoding stages (2 C+BN+LReLU), 2 C+BN+LReLU at bottleneck, skip connections (add) instead of concatenating [~(8×2)=16 million parameters]	MSE / Details not specified	Input: last iteration after all blocks Target: 3D high count reconstruction (128×128×105)	18
FBSEM-Net Mehranian and Reader [94]	**1** shared CNN for all blocks: 3 C layers each with BN+ReLU, 12 unrolled iterations [~77,000 parameters]	MSE / Adam (200)	Input: last iteration after all blocks Target: 3D high count reconstruction Cropped: (114×114×128) (for (344×344×128))	35
BCD-Net Lim *et al.* [100]	**10** convolutional autoencoders, each composed of 2 C layers and soft thresholding operator in between [~(10×4000)=40,000 parameters]	MSE / Adam (500)	Input: current output from a block Target: true activity image (200×200×112)	4

n dependent CNN

$$\boldsymbol{z}^{n+1} = F^n\left(\boldsymbol{x}^n\right). \tag{62}$$

Then this denoised image is used as a prior for a full MAP-EM reconstruction

$$\boldsymbol{x}^{n+1} = \underset{\boldsymbol{x}}{\operatorname{argmax}}\, L(\boldsymbol{x} \mid \boldsymbol{m}) - \frac{\rho}{2}\left\|\boldsymbol{x} - \boldsymbol{z}^{n+1}\right\|^2. \tag{63}$$

The method then repeats, starting again from (62). Specifically for BCD-Net, the method is proposed initially with a simple 3 layer network, although is extendable in principle to deeper architectures, such as a U-Net, as was considered by Lim *et al.*

Table IV summarizes the different deep architectures used by the various unrolled methods, and Table V summarizes the key similarities and differences between MAPEM-Net, FBSEM-Net, and BCD-Net. An advantage of the BCD-Net method, compared to the other unrolled methods, is lower demand on computational memory, as distinctly separate reconstruction and denoising training at the block level is executed without the need for a very deep single network to be in memory. In contrast, the other unrolled methods involve back-propagation through all blocks, which is memory intensive during training.

VIII. SUMMARY AND FUTURE PERSPECTIVES

After briefly reviewing the core components of PET image reconstruction and the foundations of deep learning, the various ways of integrating the data-driven benefits of deep learning into image reconstruction have been reviewed. Table I summarized four core ways in which deep learning can be integrated into the PET image reconstruction process.

Direct deep learned mappings from sinograms to images abandon all prior knowledge of physics and the noise distribution of the data, and seek instead to learn these from scratch. This has the advantage of avoiding any inaccurate modeling assumptions, but the disadvantage of entrusting these models to purely what is included in the training data only. However, if given sufficient training data, these should prove to be powerful and rapid reconstruction methods, although likely still computationally challenging for true fully 3-D reconstruction. It is notable from Table II that these methods, demonstrated in 2-D, tend to need training dataset sizes ranging from tens of thousands to hundreds of thousands of image slices (each paired with their measured data).

The synthesis approach uses deep learning for the object model only, using a deep network as a representation, then leaving the rest of the image reconstruction to follow conventional approaches. However, this requires the final reconstructed image to be the output of a network only, with the potential advantages and disadvantages this may entail in terms of what is, and is not, expressed. Alternative analysis methods use these same or similar deep learned object models not as an imposed representation, but instead as a means of analysis regularization, whereby the reconstructed image is penalized if it deviates too far from the object representation model. This could be included, for example, within a conventional L2 penalty term for a MAP reconstruction. The advantages of the synthesis and analysis approaches compared to full direct deep learning mappings include: demonstrated practicality for 3-D reconstruction, reduced need for training data (typically

TABLE V
COMPARISON OF KEY DISTINGUISHING FEATURES BETWEEN THE THREE UNROLLED METHODS: BCD-NET, MAPEM-NET, AND FBSEM-NET

METHOD	Initial image	Network different for each block?	MAPEM updates before update of prior	Use of residual image ($\boldsymbol{\mu}$) to modify the denoised prior	Training loss	Backpropagation through EM update?
BCD-Net	10 EM updates	Yes	From 1 to many	No	Block level training to fit just one ("true") reference (MSE)	No
MAPEM-Net	2 EM updates	Yes	2	Yes	MSE for end image	Yes
FBSEM-Net	10 EM updates	No	1	No	MSE for end image	Not implemented (but backprop through all blocks)

of the order of tens of 3-D images are used), and exploitation of conventional image reconstruction knowledge that we can trust (e.g., imaging physics, data corrections, and the Poisson noise model). However, their reconstruction speed will be comparable to conventional iterative methods.

Going a step further, it is possible to completely learn the regularization, not even relying on a quadratic or similar potential function to describe the penalty. For these approaches, which make even fewer assumptions regarding even *how* to regularize, it is necessary to unroll or unfold the iterative algorithm, as was shown in Fig. 14. Such approaches have similar training data needs to the synthesis/analysis group (of the order of tens of 3-D images), and similar execution times, but can be demanding for computational memory during training if they require backpropagation of gradients through all the unrolled blocks. Notably, BCD-Net avoids that need, potentially being a more practical method. Results in the literature show promise for the use of unrolled methods, but there is however now a need to compare performance between these various approaches for the same sets of training and test data (which should be as diverse as possible), preferably robustly comparing to post-reconstruction deep learning alternatives as well.

Likely future directions may include fully Bayesian deep learning [101] for PET image reconstruction, whereby not just MAP estimates are sought, but the entire posterior probability distribution. This allows uncertainty in the deep learned modeling itself to be expressed, which is useful for when high quality images are produced that may nonetheless contain uncertain features which need to be indicated to the radiologist, or specified alongside any quantitative measures of interest. The practicalities of using an image with a counterpart uncertainty image may be challenging for translation to clinical use.

Another major area of research is the need for ground truth data or high quality reference data paired with the measured data for conventional supervised learning. There will likely be a lot of potential for seeking out improved ground truth reference information, or even for development of self-supervised deep learning for image reconstruction (e.g., [102] for MRI). In these methods, rather than supplying targets/labels, instead the algorithms are provided with the knowledge of how to produce targets/labels, usually based on degradation of supplied data (such as reduced sampling, or introduction of noise) in order to recover the full input data. Furthermore, another important development is that of cycle GANs [4], which provide a powerful means of avoiding the need for matched training pairs in deep mappings. Instead, these learn, effectively unsupervised, how to map one distribution to another distribution, allowing use of pools of inputs and targets, unpaired. Cycle-consistent GANs, originally proposed in the context of image-to-image translation, could prove immensely useful in the image reconstruction context, as indeed is beginning to be the case already for MRI [103].

This present review has focused strictly on methods which involve raw PET data, primarily in the form of sinograms. As acknowledged, there has however been significant work on post-reconstruction deep learning for denoising and resolution enhancement, and perhaps these simpler approaches are more likely to be adopted at least in the shorter term. This is due to their reduced memory requirements (use of images rather than sinograms) and their apparently competitive performance with full deep-learning reconstruction methods that use the raw PET data. Recent work has shown that the relatively simple post-reconstruction methods can fare very well indeed (see again the findings with a post-reconstruction U-Net in [94], as was shown in Figs. 16 and 17 in this present review). The potential advantage of direct use of raw PET (sinogram) data (whether in direct methods or unrolled methods) perhaps is still in need of more convincing demonstration. Therefore, methods like that of Whiteley *et al.* [77] with their use of TOF backprojected images as the starting point for deep learning, do look promising in the near future.

REFERENCES

[1] S. Arridge, P. Maass, O. Oktem, and C. B. Schonlieb, "Solving inverse problems using data-driven models," *Acta Numerica*, vol. 28, pp. 1–174, May 2019.

[2] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, May 2015.

[3] D. P. Kingma and M. Welling. (2019). *An Introduction to Variational Autoencoders*. [Online]. Available: https://arxiv.org/abs/1906.02691

[4] J. Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, 2017, pp. 2242–2251.

[5] G. M. Lin *et al.*, "Deep unsupervised learning for image super-resolution with generative adversarial network," *Signal Process. Image Commun.*, vol. 68, pp. 88–100, Oct. 2018.

[6] L. Chen, P. Bentley, K. Mori, K. Misawa, M. Fujiwara, and D. Rueckert, "Self-supervised learning for medical image analysis using image context restoration," *Med. Image Anal.*, vol. 58, Dec. 2019, Art. no. 101539.

[7] I. J. Goodfellow *et al.*, "Generative adversarial nets," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, vol. 27, 2014, pp. 2672–2680.

[8] C. R. Floyd, "An artificial neural network for SPECT image reconstruction," *IEEE Trans. Med. Imag.*, vol. 10, no. 3, pp. 485–487, Sep. 1991.

[9] J. Deng, W. Dong, R. Socher, L. J. Li, K. Li, and F. F. Li, "ImageNet: A large-scale hierarchical image database," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, vols. 1–4, pp. 248–255, 2009.

[10] C. P. M. Hatt, J. Qi, and I. El Naqa, "Machine (deep) learning methods for image processing and radiomics," *IEEE Trans. Radiat. Plasma Med. Sci.*, vol. 3, no. 2, pp. 104–108, Mar. 2019.

[11] Y. Yang, J. Sun, H. B. Li, and Z. B. Xu, "Deep ADMM-net for compressive sensing MRI," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, vol. 29, 2016, pp. 10–18.

[12] B. Zhu, J. Z. Liu, S. F. Cauley, B. R. Rosen, and M. S. Rosen, "Image reconstruction by domain-transform manifold learning," *Nature*, vol. 555, no. 7697, pp. 487–492, Mar. 2018.

[13] K. Gong, C. Catana, J. Y. Qi, and Q. Z. Li, "PET image reconstruction using deep image prior," *IEEE Trans. Med. Imag.*, vol. 38, no. 7, pp. 1655–1665, Jul. 2019.

[14] I. Häggström, C. R. Schmidtlein, G. Campanella, and T. J. Fuchs, "DeepPET: A deep encoder-decoder network for directly solving the PET image reconstruction inverse problem," *Med. Imag. Anal.*, vol. 54, pp. 253–262, May 2019.

[15] D. Wu, K. Kim, and Q. Li, "Computationally efficient deep neural network for computed tomography image reconstruction," *Med. Phys.*, vol. 46, no. 11, pp. 4763–4776, Nov. 2019.

[16] W. Shao, M. G. Pomper, and Y. Du, "A learned reconstruction network for SPECT imaging," *IEEE Trans. Radiat. Plasma Med. Sci.*, early access, May 12, 2020, doi: 10.1109/TRPMS.2020.2994041.

[17] S. Ravishankar, J. C. Ye, and J. A. Fessler, "Image reconstruction: From sparsity to data-adaptive methods and machine learning," *Proc. IEEE*, vol. 108, no. 1, pp. 86–109, Jan. 2020.

[18] K. Gong, E. Berg, S. R. Cherry, and J. Y. Qi, "Machine learning in PET: From photon detection to quantitative image reconstruction," *Proc. IEEE*, vol. 108, no. 1, pp. 51–68, Jan. 2020.

[19] G. Wang, J. C. Ye, K. Mueller, and J. A. Fessler, "Image reconstruction is a new Frontier of machine learning," *IEEE Trans. Med. Imag.*, vol. 37, no. 6, pp. 1289–1296, Jun. 2018.

[20] V. Antun, F. Renna, C. Poon, B. Adcock, and A. C. Hansen, "On instabilities of deep learning in image reconstruction and the potential costs of AI," *Proc. Nat. Acad. Sci. USA*, May 2020, Art. no. 201907377. [Online]. Available: https://www.pnas.org/content/early/2020/05/08/1907377117

[21] J. Y. Qi and R. M. Leahy, "Iterative reconstruction techniques in emission computed tomography," *Phys. Med. Biol.*, vol. 51, no. 15, pp. R541–R578, Aug. 2006.

[22] A. J. Reader, and H. Zaidi, "Advances in PET image reconstruction," *PET Clin.*, vol. 2, no. 2, pp. 173–190, 2008.

[23] A. J. Reader, "The promise of new PET image reconstruction," *Physica Medica*, vol. 24, no. 2, pp. 49–56, 2008.

[24] R. M. Lewitt and S. Matej, "Overview of methods for image reconstruction from projections in emission computed tomography," *Proc. IEEE*, vol. 91, no. 10, pp. 1588–1611, Oct. 2003.

[25] C. Tsoumpas, F. E. Turkheimer, and K. Thielemans, "A survey of approaches for direct parametric image reconstruction in emission tomography," *Med. Phys.*, vol. 35, no. 9, pp. 3963–3971, Sep. 2008.

[26] A. J. Reader and J. Verhaeghe, "4D image reconstruction for emission tomography," *Phys. Med. Biol.*, vol. 59, no. 22, pp. 371–418, 2014.

[27] L. A. Shepp and Y. Vardi, "Maximum likelihood reconstruction for emission tomography," *IEEE Trans. Med. Imag.*, vol. 1, no. 2, pp. 113–122, Oct. 1982.

[28] K. Lange and R. Carson, "EM reconstruction algorithms for emission and transmission tomography," *J. Comput. Assist. Tomogr.*, vol. 8, no. 2, pp. 306–316, Apr. 1984.

[29] H. H. Barrett, D. W. Wilson, and B. M. W. Tsui, "Noise properties of the EM algorithm 1. Theory," *Phys. Med. Biol.*, vol. 39, no. 5, pp. 833–846, May 1994.

[30] A. Mehranian *et al.*, "PET image reconstruction using multi-parametric anato-functional priors," *Phys. Med. Biol.*, vol. 62, no. 15, pp. 5975–6007, Jul. 2017.

[31] M. Lustig, D. Donoho, and J. M. Pauly, "Sparse MRI: The application of compressed sensing for rapid MR imaging," *Magn. Reson. Med.*, vol. 58, no. 6, pp. 1182–1195, Dec. 2007.

[32] A. R. De Pierro, "A modified expectation maximization algorithm for penalized likelihood estimation in emission tomography," *IEEE Trans. Med. Imag.*, vol. 14, no. 1, pp. 132–137, Mar. 1995.

[33] G. Wang and J. Qi, "Penalized likelihood PET image reconstruction using patch-based edge-preserving regularization," *IEEE Trans. Med. Imag.*, vol. 31, no. 12, pp. 2194–2204, Dec. 2012.

[34] E. Levitan and G. T. Herman, "A maximum a posteriori probability expectation maximization algorithm for image-reconstruction in emission tomography," *IEEE Trans. Med. Imag.*, vol. 6, no. 3, pp. 185–192, Sep. 1987.

[35] P. Novosad and A. J. Reader, "MR-guided dynamic PET reconstruction with the kernel method and spectral temporal basis functions," *Phys. Med. Biol.*, vol. 61, no. 12, pp. 4624–4644, Jun. 21, 2016.

[36] G. Wang and J. Qi, "PET image reconstruction using kernel method," *IEEE Trans. Med. Imag.*, vol. 34, no. 1, pp. 61–71, Jan. 2015.

[37] J. Bland *et al.*, "MR-guided kernel EM reconstruction for reduced dose PET imaging," *IEEE Trans. Radiat. Plasma Med. Sci.*, vol. 2, no. 3, pp. 235–243, May 2018.

[38] A. J. Reader, F. C. Sureau, C. Comtat, R. Trebossen, and I. Buvat, "Joint estimation of dynamic PET images and temporal basis functions using fully 4D ML-EM," *Phys. Med. Biol.*, vol. 51, no. 21, pp. 5455–5474, Nov. 2006.

[39] M. Zhang, J. Zhou, X. Niu, E. Asma, W. Wang, and J. Qi, "Regularization parameter selection for penalized-likelihood list-mode image reconstruction in PET," *Phys. Med. Biol.*, vol. 62, no. 12, pp. 5114–5130, Jun. 2017.

[40] A. J. Reader and S. Ellis, "Bootstrap-optimised regularised image reconstruction for emission tomography," *IEEE Trans. Med. Imag.*, vol. 39, no. 6, pp. 2163–2175, Jun. 2020.

[41] M. Wang and W. Deng, "Deep visual domain adaptation: A survey," *Neurocomputing*, vol. 312, pp. 135–153, Oct. 2018.

[42] J. G. Colsher, "Fully 3-dimensional positron emission tomography," *Phys. Med. Biol.*, vol. 25, no. 1, pp. 103–115, 1980.

[43] P. E. Kinahan and J. G. Rogers, "Analytic 3D image-reconstruction using all detected events," *IEEE Trans. Nucl. Sci.*, vol. 36, no. 1, pp. 964–968, Feb. 1989.

[44] G. Chu and K. C. Tam, "3-dimensional imaging in positron camera using Fourier techniques," *Phys. Med. Biol.*, vol. 22, no. 2, pp. 245–265, 1977.

[45] V. V. Selivanov and R. Lecomte, "Fast PET image reconstruction based on SVD decomposition of the system matrix," *IEEE Trans. Nucl. Sci.*, vol. 48, no. 3, pp. 761–767, Jun. 2001.

[46] A. López-Montes *et al.*, " 3D PET image with pseudoinverse reconstruction," *Appl. Sci.*, vol. 10, no. 8, p. 2829, 2020.

[47] C. Bishop, *Pattern Recognition and Machine Learning*. New York, NY, USA: Plenum Press, 2006.

[48] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016.

[49] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," in *Proc. 14th Int. Conf. Artif. Intell. Stat.*, vol. 15, 2011, pp. 315–323.

[50] M. A. Kramer, "Nonlinear principal component analysis using autoassociative neural networks," *AIChE J.*, vol. 37, no. 2, pp. 233–243, 1991.

[51] B. B. Y. Le Cun, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel "Handwritten digit recognition with a back-propagation network," in *Proc. Adv. Neural Inf. Process. Syst.*, 1990, pp. 396–404.

[52] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2323, Nov. 1998.

[53] D. Rigie, G. Schramm, T. Vahle, T. Shepherd, J. Nuyts, and F. Boada, "Approximating MRI-based anatomically guided PET reconstruction with a convolutional neural network," in *Proc. IEEE Nucl. Sci. Symp. Med. Imag. Conf. Proc. (NSS/MIC)*, Sydney, NSW, Australia, 2018, pp. 1–3.

[54] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural Netw.*, vol. 2, no. 5, pp. 359–366, 1989.

[55] C. O. da Costa Luis and A. J. Reader, "Micro-networks for robust MR-guided low count PET imaging," *IEEE Trans Radiat Plasma Med Sci*, early access, Apr. 8, 2020, doi: 10.1109/TRPMS.2020.2986414.

[56] J. C. Ye and W. K. Sung, "Understanding geometry of encoder–decoder CNNs," in *Proc. 36th Int. Conf. Mach. Learn.*, 2019, pp. 1–26.

[57] M. S. Tahaei and A. J. Reader, "Patch-based image reconstruction for PET using prior-image derived dictionaries," *Phys. Med. Biol.*, vol. 61, no. 18, pp. 6833–6855, Sep. 2016.

[58] Y. C. Pu *et al.*, "Variational autoencoder for deep learning of images, labels and captions," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, vol. 29, 2016, pp. 2352–2360.

[59] G. Yang *et al.*, "DAGAN: Deep de-aliasing generative adversarial networks for fast compressed sensing MRI reconstruction," *IEEE Trans. Med. Imag.*, vol. 37, no. 6, pp. 1310–1321, Jun. 2018.

[60] Z. Hu *et al.*, "DPIR-Net: Direct PET image reconstruction based on the Wasserstein generative adversarial network," *IEEE Trans. Radiat. Plasma Med. Sci.*, early access, May 19, 2020, doi: 10.1109/TRPMS.2020.2995717.

[61] Y. Wang *et al.*, "3D conditional generative adversarial networks for high-quality PET image estimation at low dose," *Neuroimage*, vol. 174, pp. 550–562, Jul. 2018.

[62] S. Kaplan and Y. M. Zhu, "Full-dose PET image estimation from low-dose PET image using deep learning: A pilot study," *J. Digit. Imag.*, vol. 32, no. 5, pp. 773–778, Oct. 2019.

[63] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional networks for biomedical image segmentation," in *Medical Image Computing and Computer-Assisted Intervention* (Lecture Notes in Computer Science), vol. 9351, N. Navab, J. Hornegger, W. Wells, A. Frangi, Eds. Cham, Switzerland: Springer, 2015. [Online]. Available: https://doi.org/10.1007/978-3-319-24574-4_28

[64] J. C. Ye, Y. Han, and E. Cha, "Deep convolutional framelets: A general deep learning framework for inverse problems," *SIAM J. Imag. Sci.*, vol. 11, no. 2, pp. 991–1048, 2018.

[65] Y. Han and J. C. Ye, "Framing U-net via deep convolutional framelets: Application to sparse-view CT," *IEEE Trans. Med. Imag.*, vol. 37, no. 6, pp. 1418–1429, Jun. 2018.

[66] L. Bottou, F. E. Curtis, and J. Nocedal, "Optimization methods for large-scale machine learning," *SIAM Rev.*, vol. 60, no. 2, pp. 223–311, 2018.

[67] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proc. Int. Conf. Learn. Represent.*, 2015, pp. 1–14.

[68] K. Gong, J. Guan, C. Liu, and J. Qi, "PET image denoising using a deep neural network through fine tuning," *IEEE Trans. Radiat. Plasma Med. Sci.*, vol. 3, no. 2, pp. 153–161, Mar. 2019.

[69] K. T. Chen *et al.*, "Generalization of deep learning models for ultra-low-count amyloid PET/MRI using transfer learning," *Eur. J. Nucl. Med. Mol. Imag.*, early access, Jun. 13, 2020, doi: 10.1007/s00259-020-04897-6.

[70] P. Novosad, V. Fonov, and D. L. Collins, "Accurate and robust segmentation of neuroanatomy in T1-weighted MRI by combining spatial priors with deep convolutional neural networks," *Human Brain Map.*, vol. 41, no. 2, pp. 309–327, Feb. 2020.

[71] Y. Wang *et al.*, "3D auto-context-based locality adaptive multi-modality GANs for PET synthesis," *IEEE Trans. Med. Imag.*, vol. 38, no. 6, pp. 1328–1339, Jun. 2019.

[72] J. A. Fessler, "Optimization methods for magnetic resonance image reconstruction: Key models and optimization algorithms," *IEEE Signal Process. Mag.*, vol. 37, no. 1, pp. 33–40, Jan. 2020.

[73] W. Lu *et al.*, "An investigation of quantitative accuracy for deep learning based denoising in oncological PET," *Phys. Med. Biol.*, vol. 64, no. 16, Aug. 2019, Art. no. 165019.

[74] L. Zhou, J. D. Schaefferkoetter, I. W. K. Tham, G. Huang, and J. Yan, "Supervised learning with cyclegan for low-dose FDG PET image denoising," *Med. Image Anal.*, vol. 65, Jul. 2020, Art. no. 101770.

[75] K. Erlandsson, A. J. Reader, M. A. Flower, and R. J. O. Joint, "A new 3D backprojection and filtering method for pet using all detected events," *IEEE Trans. Nucl. Sci.*, vol. 45, no. 3, pp. 1183–1188, Jun. 1998.

[76] S. Matej, S. Surti, S. Jayanthi, M. E. Daube-Witherspoon, R. M. Lewitt, and J. S. Karp, "Efficient 3-D TOF PET reconstruction using view-grouped histo-images: DIRECT-direct image reconstruction for TOF," *IEEE Trans. Med. Imag.*, vol. 28, no. 5, pp. 739–751, May 2009.

[77] W. Whiteley, V. Panin, C. Zhou, J. Cabello, D. Bharkhada, and J. Gregor, "FastPET: Near real-time PET reconstruction from histo-images using a neural network," in *Proc. IEEE NSS MIC Conf. Rec.*, 2019, pp. 1–14.

[78] Z. Liu, H. Chen, and H. Liu, "Deep learning based framework for direct reconstruction of PET images," in *Medical Image Computing and Computer Assisted Intervention* (Lecture Notes in Computer Science), vol. 11766, D. Shen *et al.* Eds. Cham, Switzerland: Springer, 2019. [Online]. Available: https://doi.org/10.1007/978-3-030-32248-9_6

[79] W. Whiteley and J. Gregor, "Efficient neural network image reconstruction from raw data using a radon inversion layer," in *Proc. IEEE NSS MIC Conf. Rec.*, 2019, pp. 1–2.

[80] C. Comtat *et al.*, "OSEM-3D reconstruction strategies for the ECAT HRRT," in *Proc. IEEE Nucl. Sci. Symp. Conf. Rec.*, vols. 1–7, 2004, pp. 3492–3496.

[81] M. E. Daube-Witherspoon and G. Muehllehner, "Treatment of axial data in three-dimensional PET," *J. Nucl. Med.*, vol. 28, no. 11, pp. 1717–1724, Nov. 1987.

[82] K. Gong *et al.*, "Iterative PET Image Reconstruction Using Convolutional Neural Network Representation," *IEEE Trans. Med. Imag.*, vol. 38, no. 3, pp. 675–685, Mar. 2019.

[83] D. Ulyanov, A. Vedaldi, and V. Lempitsky, "Deep image prior," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2018, pp. 9446–9454.

[84] K. Gong, C. Catana, J. Y. Qi, and Q. Z. Li, "Direct patlak reconstruction for low dose dynamic PET using unsupervised deep learning," *J. Nucl. Med.*, vol. 60, p. 575, May 2019.

[85] T. Yokota, K. Kawai, M. Sakata, Y. Kimura, and H. Hontani, "Dynamic PET image reconstruction using nonnegative matrix factorization incorporatedwith deep image prior," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2019.

[86] Z. Xie, R. Baikejiang, K. Gong, X. Zhang, and J. Qi, "Generative adversarial networks based regularized image reconstruction for PET," in *Proc. SPIE 15th Int. Meeting Fully Three Dimensional Image Reconstruct. Radiol. Nucl. Med.*, 2019, Art. no. 11072.

[87] N. Xie *et al.* (2019). *Penalized-Likelihood PET Image Reconstruction Using 3D Structural Convolutional Sparse Coding*. [Online]. Available: https://arxiv.org/abs/1912.07180

[88] K. Kim *et al.*, "Penalized PET reconstruction using deep learning prior and local linear fitting," *IEEE Trans. Med. Imag.*, vol. 37, no. 6, pp. 1478–1487, Jun. 2018.

[89] M. G. Poirot *et al.*, "Physics-informed deep learning for dual-energy computed tomography image processing," *Sci. Rep.*, vol. 9, no. 1, Nov. 2019, Art. no. 17709.

[90] K. Gregor and Y. LeCun, "Learning fast approximations of sparse coding," in *Proc. 27th Int. Conf. Int. Conf. Mach. Learn.*, 2010, pp. 399–406.

[91] K. Hammernik *et al.*, "Learning a variational network for reconstruction of accelerated MRI data," *Magn. Reson. Med.*, vol. 79, no. 6, pp. 3055–3071, Jun. 2018.

[92] K. Gong *et al.*, "EMNet: An unrolled deep neural network for PET image reconstruction," in *Proc. SPIE Med. Imag. Phys. Med. Imag.*, Mar. 2019. [Online]. Available: https://doi.org/10.1117/12.2513096

[93] W. Gong *et al.*, "MAPEM-Net: An unrolled neural network for Fully 3D PET image reconstruction," in *Proc. 15th Int. Meeting Fully Three Dimensional Image Reconstruct. Radiol. Nucl. Med.*, 2019, Art. no. 1107200.

[94] A. Mehranian and A. J. Reader, "Model-based deep learning PET image reconstruction using forward-backward splitting expectation maximisation," *IEEE Trans. Radiat. Plasma Med. Sci.*, early access, Jun. 23, 2020, doi: 10.1109/TRPMS.2020.3004408.

[95] P. L. Combettes and J.-C. Pesquet, "Proximal splitting methods in signal processing," in *Fixed-Point Algorithms for Inverse Problems in Science and Engineering*, H. H. Bauschke, R. S. Burachik, P. L. Combettes, V. Elser, D. R. Luke, and H. Wolkowicz, Eds. New York, NY, USA: Springer, 2011, pp. 185–212.

[96] A. R. De Pierro, "On the relation between the ISRA and the EM algorithm for positron emission tomography," *IEEE Trans. Med. Imag.*, vol. 12, no. 2, pp. 328–333, Jun. 1993.

[97] E. Levitan and G. T. Herman, "A maximum a posteriori probability expectation maximization algorithm for image reconstruction in emission tomography," *IEEE Trans. Med. Imag.*, vol. 6, no. 3, pp. 185–192, Sep. 1987.

[98] I. Y. Chun and J. A. Fessler, "Deep BCD-net using identical encoding-decoding CNN structures for iterative image recovery," in *Proc. IEEE 13th Image Video Multidimensional Signal Process. Workshop (IVMSP)*, 2018, pp. 1–5.

[99] H. Lim, Z. Huang, J. A. Fessler, Y. K. Dewaraja, and I. Y. Chun, "Application of trained Deep BCD-Net to iterative low-count PET image reconstruction," in *Proc. IEEE Nucl. Sci. Symp. Med. Imag. Conf. Proc. (NSS/MIC)*, Sydney, NSW, Australia, 2018, pp. 1–4.

[100] H. Lim, I. Y. Chun, Y. K. Dewaraja, and J. A. Fessler, "Improved low-count quantitative PET reconstruction with an iterative neural network," *IEEE Trans. Med. Imag.*, early access, May 29, 2020, doi: 10.1109/TMI.2020.2998480.

[101] A. Kendall and Y. Gal, "What uncertainties do we need in Bayesian deep learning for computer vision?" in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, 2017, pp. 5574–5584.

[102] B. Yaman, S. Hosseini, S. Moeller, J. Ellermann, K. Ugurbil, and M. Akcakaya, "Self-supervised physics-based deep learning MRI reconstruction without fully-sampled data," in *Proc. IEEE Int. Symp. Biomed. Imag.*, 2020, pp. 921–925.

[103] G. Oh, B. Sim, and J.-C. Ye, "Unsupervised learning for compressed sensing MRI using cycleGAN," in *Proc. IEEE Int. Symp. Biomed. Imag.*, 2020, pp. 1082–1085.

[104] M. A. Belzunce and A. J. Reader, "Technical Note: Ultra high-resolution radiotracer-specific digital pet brain phantoms based on the BigBrain atlas," *Med. Phys*. Accessed: May 5, 2020. [Online]. Available: https://doi.org/10.1002/mp.14218

[105] C. Cocosco, V. Kollokian, R.-S. Kwan, and A. Evans, "BrainWeb: Online interface to a 3D MRI simulated brain database," *NeuroImage*, vol. 5, no. 4, pp. S424–S425, 1997.