RECURRENT INFERENCE MACHINES FOR SOLVING INVERSE PROBLEMS

Patrick Putzky & Max Welling
Informatics Institute
University of Amsterdam
{pputzky,m.welling}@uva.nl

ABSTRACT

Inverse problems are typically solved by first defining a model and then choosing an inference procedure. With this separation of modeling from inference, inverse problems can be framed in a modular way. For example, variational inference can be applied to a broad class of models. The modularity, however, typically goes away after model parameters have been trained under a chosen inference procedure. During training, model and inference often interact in a way that the model parameters will ultimately be adapted to the chosen inference procedure, posing the two components inseparable after training. But if model and inference become inseperable after training, why separate them in the first place?

We propose a novel learning framework which abandons the dichotomy between model and inference. Instead, we introduce *Recurrent Inference Machines (RIM)*, a class of recurrent neural networks (RNN), that directly learn to solve inverse problems.

We demonstrate the effectiveness of RIMs in experiments on various image reconstruction tasks. We show empirically that RIMs exhibit the desirable convergence behavior of classical inference procedures, and that they can outperform state-of-the-art methods when trained on specialized inference tasks.

Our approach bridges the gap between inverse problems and deep learning, providing a framework for fast progression in the field of inverse problems.

1 INTRODUCTION

Inverse Problems are a broad class of problems which can be encountered in all scientific disciplines, from the natural sciences to engineering. The task in inverse problems is to reconstruct a signal from observations that are subject to a known (or inferred) corruption process known as the forward model. A typical example of an inverse problem is the linear measurement problem

$$\mathbf{y} = \mathbf{A}\mathbf{x} + \mathbf{n}, \tag{1}$$

where $\mathbf{x}$ is the signal of interest, $\mathbf{A}$ is an $m \times d$ corruption matrix, $\mathbf{n}$ is an additive noise vector, and $\mathbf{y}$ is the actual measurement. If $\mathbf{A}$ is a wide matrix such that $m \ll d$, this problem is typically ill-posed. Many signal reconstruction problems can be phrased in terms of the linear measurement problem such as image denoising, super-resolution, deconvolution and so on. The general form of $\mathbf{A}$ typically defines the problem class. If $\mathbf{A}$ is an identity matrix the problem is a denoising problem, while in tomography $\mathbf{A}$ represents a Fourier transform and a consecutive sub-sampling of the Fourier coefficients.

Inverse problems are often formulated as an optimization problem of the form

$$\min_{\mathbf{x}} d(\mathbf{y}, \mathbf{A}\mathbf{x}) + \lambda R(\mathbf{x}), \tag{2}$$

where $d(\mathbf{y}, \mathbf{A}\mathbf{x})$ is the data fidelity term that enforces $\mathbf{x}$ to satisfy the observations $\mathbf{y}$, and $R(\mathbf{x})$ is a regularization term which restricts the solution to comply with a predefined model over $\mathbf{x}$.

The difficulties that arise in this framework are two-fold: (1) it is difficult to choose $R(\mathbf{x})$ such that it is an appropriate model for complex signals such as natural images, and (2) even under a well chosen $R(\mathbf{x})$ the optimization procedure might become difficult.

Compressed sensing approaches give up on a versatile $R(\mathbf{x})$ in order to define a convex optimization procedure. The idea is that the signal $\mathbf{x}$ has a sparse representation in some basis Ψ such that $\mathbf{x} = \Psi\mathbf{u}$ and that the optimization problem can be rephrased as

$$\min_{\mathbf{u}} d(\mathbf{y}, \mathbf{A}\Psi\mathbf{u}) + \lambda \left\|\mathbf{u}\right\|_1, \tag{3}$$

where $\left\|\cdot\right\|_1$ is the sparsity inducing L1-norm (Donoho, 2006a). Under certain classes of $d(\mathbf{y}, \mathbf{A}\Psi\mathbf{u})$ such as quadratic errors the optimization problem becomes convex. Results from the compressed sensing literature offer provable bounds on the reconstruction performance for sparse signals of this form (Candès et al., 2006; Donoho, 2006b). The basis Ψ can also be learned from data (Aharon et al., 2006; Elad & Aharon, 2006).

Other approaches interpret equation (2) in terms of probabilities such that finding the solution is a matter of performing maximum a posteriori (MAP) estimation (Figueiredo et al., 2007). In those cases $d(\mathbf{y}, \mathbf{A}\Psi\mathbf{u})$ takes the form of a log-likelihood and $R(\mathbf{x})$ takes the form of a parametric log-prior $\log p_\theta(\mathbf{x})$ over variable $\mathbf{x}$ such that the minimization becomes:

$$\max_{\mathbf{x}} \log p(\mathbf{y}|\mathbf{A}, \mathbf{x}) + \log p_\theta(\mathbf{x}). \tag{4}$$

This allows for more expressiveness of $R(\mathbf{x})$ and for the possibility of learning the prior $p_\theta(\mathbf{x})$ from data. However, with more expressive priors optimization will become more difficult as well. In fact, only for a few trivial prior-likelihood pairs will inference remain convex. In practice one often has to resort to approximations of the objective and to approximate double-loop algorithms in order to allow for scalable inference (Nickisch & Seeger, 2009; Zoran & Weiss, 2011).

In this work we take a radically different approach to solving inverse problems. We move away from the idea that it is beneficial to separate learning a prior (regularizer) from the optimization to do the reconstruction. The usual thinking is that this separation allows for greater modularity and the possibility to interchange one of these two complementary components in order to build new algorithms. In practice however, we observe that the optimization procedure almost always has to be adapted to the model choice to achieve good performance (Aharon et al., 2006; Elad & Aharon, 2006; Nickisch & Seeger, 2009; Zoran & Weiss, 2011). In fact, it is well known that the optimization procedure used for training should match the one used during testing because the model has adapted itself to perform well under that optimization procedure (Kumar et al., 2005; Wainwright, 2006).

What we need is a single framework which allows us to backpropagate through the optimization procedure when we learn the free parameters. Hence, We propose to look at inverse problems as a direct mapping from observations to estimated signal,

$$\hat{\mathbf{x}} = f_\phi(\mathbf{A}, \mathbf{y}) \tag{5}$$

where $\hat{\mathbf{x}}$ is an estimate of signal $\mathbf{x}$ from observations $(\mathbf{A}, \mathbf{y})$. Here we define ϕ as a set of learnable parameters which define the inference algorithm as well as constraints on $\mathbf{x}$. The goal is thus to define map whose parameters are directly optimized for solving the inverse problem itself. It has the benefits of both having high expressive power (if the map f_ϕ is complex enough) as well as being fast at inference time.

This paradigm shift allows us to learn and combine the effect of a prior, the reconstruction fidelity and an inference method without the need to explicitly define the functional form of all components. The whole procedure is simply interpreted as a single RNN. As a result, there is no need for sparsity assumptions, the introduction of model constraints to allow for convexity, or even for double-loop algorithms (Gregor & LeCun, 2010). In fact the proposed framework allows for use of current deep learning approaches which have high expressive power without trading off scalability. It further allows us to move all the manual parameter tuning - which is still common in traditional approaches (Zoran & Weiss, 2011) - away from the inference phase and into the learning phase. We believe this framework can be an important asset to introduce deep learning into the domain of inverse problems.

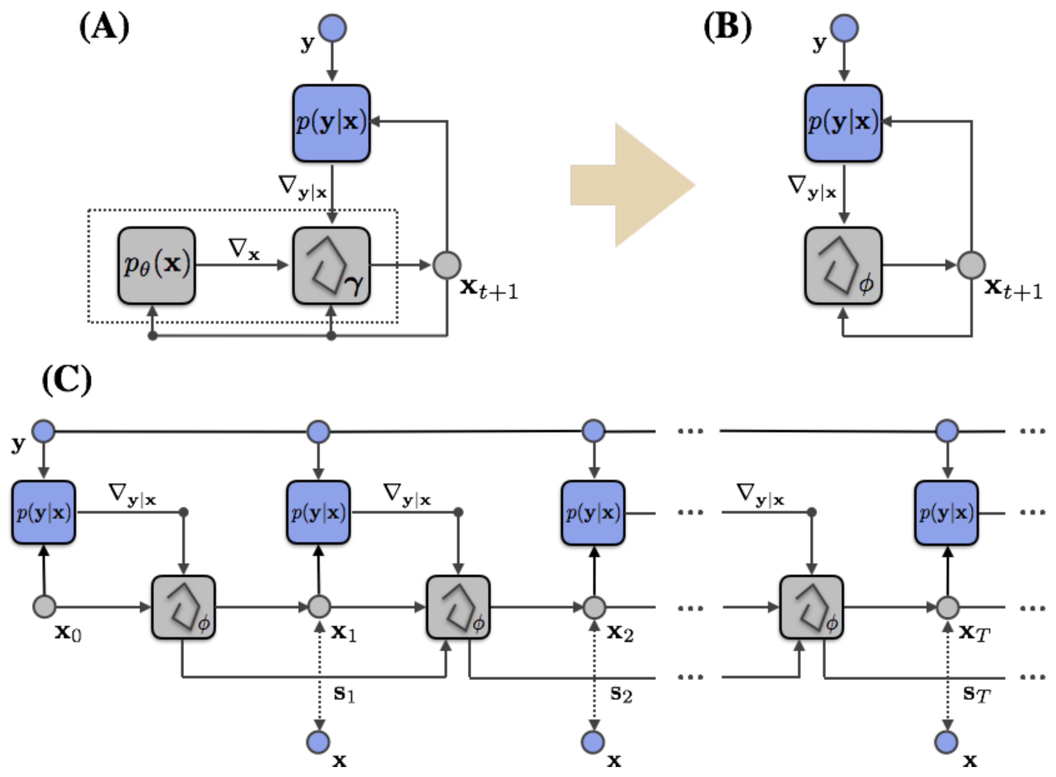

Figure 1: **(A)** *Graphical illustration of the recurrent structure of MAP estimation* (compare equation (6)). The three boxes represent likelihood model $p(\mathbf{y}|\mathbf{x})$ ($\mathbf{A}$ omitted), prior $p_\theta(\mathbf{x})$, and update function Γ, respectively. In each iteration, likelihood and prior collect the current estimate of $\mathbf{x}$, to send a gradient to update function Γ (see text). Γ then produces a new estimate of $\mathbf{x}$. Typically, prior $p_\theta(\mathbf{x})$ and update function Γ are modeled as two distinct model components. Here they are both depicted in gray boxes because they each represent model internal information which we wish to be transferable between different observations, i.e. they are observation independent. Likelihood term $p(\mathbf{y}|\mathbf{x})$ is depicted in blue to emphasize it as a model extrinsic term, some aspects of the likelihood term can change from one observation to the other (such as matrix $\mathbf{A}$). The likelihood term is observation-dependent. **(B)** *Model simplification.* The central insight of this work is to merge prior $p_\theta(\mathbf{x})$ and update function Γ into one model with trainable parameters ϕ. The model then iteratively produces new estimates through feedback from likelihood model $p(\mathbf{y}|\mathbf{x})$ and previous updates. **(C)** *A Recurrent Inference Machine unrolled in time.* Here we have added an additional state variable which represents information that is carried over time, but is not directly subjected to constraints through the likelihood term $p(\mathbf{y}|\mathbf{x})$. During training, estimates at each time step are subject to an error signal from the ground truth signal $\mathbf{x}$ (dashed two-sided arrows) in order to perform backpropagation. The intermittent error signal will force the model to perform well as soon as possible during iterations. At test time, there is no error signal from $\mathbf{x}$.

2 Recurrent Inference Machines

The goal of this work is to find an inverse model as described in equation (5). Often, however, it will be intractable to find (5) directly, even with modern non-linear function approximators. For high-dimensional $\mathbf{y}$ and $\mathbf{x}$, which are typically considered in inverse problems, it will simply not be possible to fit matrix $\mathbf{A}$ into memory explicitly, but instead matrix $\mathbf{A}$ will be replaced by an operator that acts on $\mathbf{x}$. An example is the Discrete Fourier Transform (DFT). Instead of using a Fourier matrix which is quadratic in the size of $\mathbf{x}$, DFTs are typically performed using the Fast Fourier Transform (FFT) algorithm which reduces computational cost and memory consumption significantly. The use of operators, however, does not allow us to feed $\mathbf{A}$ into (5) anymore, but instead we will have to resort to an iterative approach that alternates between updates of $\mathbf{x}$ and evaluation of $\mathbf{Ax}$. This is precisely what is typically done in gradient-based inference methods, and we will motivate our framework from there.

2.1 Gradient-based inference

Recall from equation (4) that inverse problems can be interpreted in terms of probability such that optimization is an iterative approach to MAP inference. In its most simple form each consecutive estimate of $\mathbf{x}$ is then computed through a recursive function of the form

$$\mathbf{x}_{t+1} = \mathbf{x}_t + \gamma_t \nabla \Big(\log p\left(\mathbf{y}|\mathbf{A},\mathbf{x}\right) + \log p_\theta\left(\mathbf{x}\right) \Big)\left(\mathbf{x}_t\right) \tag{6}$$

where we make use of the fact that $p(\mathbf{x}|\mathbf{A},\mathbf{y}) \propto p(\mathbf{y}|\mathbf{A},\mathbf{x})p_\theta(\mathbf{x})$ and γ_t is the step size or learning rate at iteration t. Further, $\mathbf{A}$ is a (partially-)observable covariate, $p(\mathbf{y}|\mathbf{A},\mathbf{x})$ is the likelihood function for a given inference problem, and $p_\theta\left(\mathbf{x}\right)$ is a prior over signal $\mathbf{x}$. In many cases where either the likelihood term or the prior term deviate from standard models, optimization will not be convex. In constrast, the approach presented in this work is completely freed from ideas about convexity, as will be shown in the next section.

2.2 Recurrent function definition

The central insight of this work is that update equation (6) can be generalized such that

$$\mathbf{x}_{t+1} = \mathbf{x}_t + g_\phi(\nabla_{\mathbf{y}|\mathbf{x}}, \mathbf{x}_t) \tag{7}$$

where we denote $\nabla \log p(\mathbf{y}|\mathbf{A},\mathbf{x})(\mathbf{x}_t)$ by $\nabla_{\mathbf{y}|\mathbf{x}}$ for readability and ϕ is a set of learnable parameters that govern the updates of $\mathbf{x}$. In this representation, prior parameters θ and learning rate parameters $\boldsymbol{\gamma}$ have been merged into one set of trainable parameters ϕ.

To recover the original update equation (6), $g_\phi(\nabla_{\mathbf{y}|\mathbf{x}}, \mathbf{x}_t)$ is written as

$$g_\phi(\nabla_{\mathbf{y}|\mathbf{x}}, \mathbf{x}_t) = \gamma_t \left(\nabla_{\mathbf{y}|\mathbf{x}} + \nabla_{\mathbf{x}}\right) \tag{8}$$

where we make use of $\nabla_{\mathbf{x}}$ to denote $\nabla \log p_\theta(\mathbf{x})(\mathbf{x}_t)$. It will be useful to dissect the terms on the right-hand side of (8) to make sense of the usefulness of the modification.

First notice, that in equation (6) we never explicitly evaluate the prior, but only evaluate its gradient in order to perform updates. If never used, learning a prior appears to be unnecessary, and instead it appears more reasonable to directly learn a gradient function $\nabla_{\mathbf{x}} = f_\theta(\mathbf{x}_t) \in \mathbb{R}^d$. The advantage of working solely with gradients is that they do no require the evaluation of an (often) intractable normalization constant of $p_\theta(\mathbf{x})$.

A second observation is that the step sizes γ_t are usually subject to either a chosen schedule or chosen through a deterministic algorithm such as a line search. That means the step sizes are always chosen according to a predefined model Γ. Interestingly, this model is usually not learned. In order to make inference faster and improve performance we suggest to learn the model Γ as well.

In (7) we have made the prior $p_\theta(\mathbf{x})$ and the the step size model Γ implicit in function $g_\phi(\nabla_{\mathbf{y}|\mathbf{x}}, \boldsymbol{\eta}_t)$. We explicitly keep $\nabla_{\mathbf{y}|\mathbf{x}}$ as an input to (7) because - as opposed to Λ and $p_\theta(\mathbf{x})$ - it represents extrinsic information that is injected into the model. It allows for changes in the likelihood model $p(\mathbf{y}|\mathbf{x})$ without the need to retrain parameters ϕ of the inference model g_ϕ. Figure 1 gives a visual summary of the insights from this section.

2.3 Output Constraints

In many problem domains the range of values for variable $\mathbf{x}$ is naturally constraint. For example, images typically have pixels with strictly positive values. In order to model this constraint we make use of nonlinear link functions as they are typically used in neural networks, such that

$$\mathbf{x} = \Psi(\boldsymbol{\eta}) \tag{9}$$

where $\Psi(\cdot)$ is any differentiable link function and $\boldsymbol{\eta}$ is the space in which RIMs iterate such that update equation (7) is replaced by

$$\boldsymbol{\eta}_{t+1} = \boldsymbol{\eta}_t + g_\phi(\nabla_{\mathbf{y}|\boldsymbol{\eta}}, \boldsymbol{\eta}_t) \tag{10}$$

As a result $\mathbf{x}$ can be constraint to a certain range of values through $\Psi(\cdot)$, whereas iterations are performed in the unconstrained space of $\boldsymbol{\eta}$

2.4 Recurrent Networks

A useful extension of (7) is to introduce a latent state variable $\mathbf{s}_t$ into the procedure. This latent variable is typically used as a utility in recurrent neural networks to learn temporal dependencies in data processing. With an additional latent variable the update equations become

$$\boldsymbol{\eta}_{t+1} = \boldsymbol{\eta}_t + h_\phi\left(\nabla_{\mathbf{y}|\boldsymbol{\eta}}, \boldsymbol{\eta}_t, \mathbf{s}_{t+1}\right) \tag{11}$$

$$\mathbf{s}_{t+1} = h^*_\phi\left(\nabla_{\mathbf{y}|\boldsymbol{\eta}}, \boldsymbol{\eta}_t, \mathbf{s}_t\right) \tag{12}$$

where $h^*_\phi(\cdot)$ is the update model for state variable $\mathbf{s}$. The variable $\mathbf{s}$ will allow the procedure to have memory in order to track progression, curvature, approximate a preconditioning matrix $\mathbf{T}_t$ (such as in BFGS) and determine a stopping criterion among other things. The concept of a temporal memory is quite limited in classical inference methods, which will allow RIMs to have a potential advantage over these methods.

2.5 Training

In order to learn a step-wise inference procedure it will be necessary to simulate the inference steps during training. I.e. during training, an RIM will perform a number of inference steps T. At each step the model will produce a prediction as depicted in figure Figure 1. Each of those predictions is then subject to a loss, which encourages the model to produce predictions that improve over time. In it's simplest form we can define a loss which is simply a weighted sum of the individual prediction losses at each time step such that

$$\mathcal{L}^{total}(\phi) = \sum_{t=1}^{T} w_t \mathcal{L}(\mathbf{x}_t(\phi), \mathbf{x}) \tag{13}$$

is the total loss. Here, $\mathcal{L}(\cdot)$ is a base loss function such as the mean square error, w_t is a positive scalar and $\mathbf{x}_t(\phi)$ is a prediction at time t. In this work we follow Andrychowicz et al. (2016) in setting $w_t = 1$ for all time steps.

3 Related Work

The RIM framework can be seen as an auto-encoder framework in which only the decoder is trained, whereas the encoder is given by a known corruption process. In terms of the training procedure this makes RIMs very similar to denoising auto-encoders (Vincent et al., 2008). Though initially with the objective of regularization in mind, denoising auto-encoders have been shown to be effectively used as generative models (Vincent et al., 2010). The difference of RIMs to denoising auto-encoders and also more recently developed auto-encoders such as Kingma & Welling (2014); Rezende et al. (2014) is that RIMs enforce coupling between encoder and decoder both, during training and test time. In it's typical form, decoder and encoder of an auto-encoder are only coupled during training time, while there is no information flow during test time (Kingma & Welling, 2014; Rezende et al., 2014; Vincent et al., 2008; 2010). An exception is the work from Gregor et al. (2016) which is conceptually strongly related to RIMs. There, an RNN model is used to generate static data by drawing on a fixed canvas. An error signal is propagated throughout the generation process.

There have been approaches in the past which aim to formulate a framework in which an inference procedure is learned. One of the best known frameworks is LISTA (Gregor & LeCun, 2010) which aims to learn a model that reconstructs sparse codes from data. LISTA models try to fit into the classical framework of doing inference as described in 1, whereas RIMs are completely removed from assumptions about sparsity. A recent paper by Andrychowicz et al. (2016) aims to train RNNs as optimizers for non-convex optimization problems. Though introduced with a different intention, RIMs can be seen as a generalization of this approach, in which the model - in addition to the gradient information - is aware about the absolute position of a prediction in variable space(see equation (7)).

4 Experimental Results

We evaluate our method on various kinds of image restoration tasks which can each be formulated in terms of linear measurement problems as described in equation (1). We first analyze the properties of our proposed method on a set of restoration tasks from random projections. Later we compare our model on two well known image restoration tasks: image denoising and image super-resolution.

4.1 Models

If not specified otherwise we use the same RNN architecture for all experiments presented in this work. The chosen RNN consists of three convolutional hidden layers and a final convolutional output layer. All convolutional filters were chosen to be of size 3 x 3 pixels. The first hidden layer consists of convolutions with stride 2 (64 features), subsequent batch normalization and a tanh nonlinearity. The second hidden layer represents the RNN part of the model. We chose a gated recurrent unit (GRU) (Chung et al., 2014) with 256 features. The third hidden layer is a transpose convolution layer with 64 features which aims to recover the original image dimensions of the signal, followed again by a batch normalization layer and a tanh nonlinearity. All models have been trained on a fixed number of iterations of 20 steps. All methods were implemented in Tensorflow[1].

4.2 Data

All experiments were run on the BSD-300 data set (Martin et al., 2001)[2]. For training we extracted patches of size 32 x 32 pixels with stride 4 from the 200 training images available in the data set. In total this amounts to a data set of about 400 thousand image patches with highly redundant information. All models were trained over only two epochs, i.e. each unique image patch was seen by a model only twice during training. Validation was performed on a held-out data set of 1000 image patches.

For testing we either used the whole test set of 100 images from BSDS-300 or we used only a subset of 68 images which was introduced by Roth & Black (2005) and which is commonly used in the image restoration community [3].

4.3 Image Restoration

All tasks addressed in this work assume a linear measurement problem of the form as described in equation (1) with additive (isotropic) Gaussian noise. In this case the gradient of the likelihood takes the form

$$\nabla_{\mathbf{y}|\mathbf{x}} = \frac{1}{\sigma^2}\mathbf{A}^T(\mathbf{y} - \mathbf{A}\mathbf{x}) \tag{14}$$

where σ^2 is the noise variance. For very small σ this gradient diverges. In order to make the gradient more stable also for small σ we chose to rewrite it as

$$\nabla_{\mathbf{y}|\mathbf{x}} = \frac{1}{\sigma^2 + \epsilon}\mathbf{A}^T(\mathbf{y} - \mathbf{A}\mathbf{x}) \tag{15}$$

where $\epsilon = \text{softplus}(\phi_\epsilon)$ and ϕ_ϵ is a trainable parameter. As a link function Ψ (see (9)) we chose the logistic sigmoid nonlinearity[4] and we used the mean square error as training loss.

4.4 Multi-task learning with Random Projections

To analyze the properties of our proposed framework in terms of convergence and to test whether all components of the model are useful, we first trained the model to reconstruct image patches from noisy random projections of grayscale image patches. We consider three types of random projection matrices: (1) Gaussian ensembles with elements drawn from a standard normal distribution, (2) binary ensembles with entries of values $\{-1, 1\}$ drawn from a Bernulli distribution with $p = 0.5$, and (3) Fourier ensembles with randomly sampled rows from a Fourier matrix (see Donoho (2006b)).

We trained three models on these tasks: (1) a Recurrent Inference Machine (RIM) as described in 2, (2) a gradient-descent network (GDN) which does not use the current estimate as an input (compare

[1]https://www.tensorflow.org

[2]https://www2.eecs.berkeley.edu/Research/Projects/CS/vision/bsds/

[3]http://www.visinf.tu-darmstadt.de/vi_research/code/foe.en.jsp

[4]All training data was rescaled to be in the range $[0, 1]$

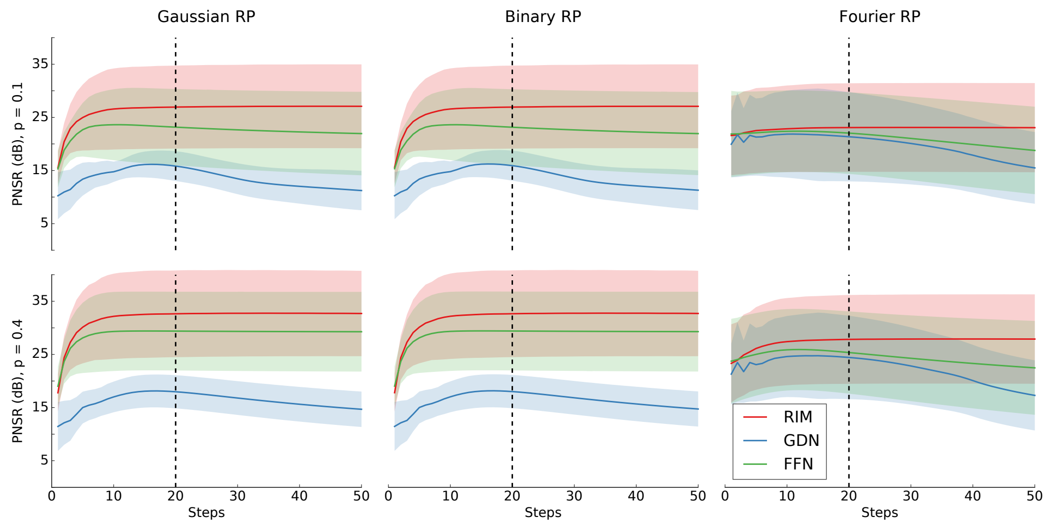

Figure 2: *Reconstruction performance over time on random projections.* Shown are results of the three reconstruction tasks from random projections (see text) on 5000 random patches from the BSD-300 test set. Value of p represent the the reduction in dimensionality through the random projection. Noise standard deviation was chosen to be $\sigma = 1$. Solid lines correspond to the mean peak signal-to-noise-ration (PSNR) over time, and shaded areas correspond to one standard deviation around the mean. Vertical dashed lines mark the last time step that was used during training.

Andrychowicz et al. (2016)), and (3) a feed-forward network (FFN) which uses the same inputs as the RIM but where we replaced the GRU unit with a ReLu layer in order to remove state-dependence. Model (2) and (3) are simplifications of RIM in order to test the influence of each of the removed model components on prediction performance.

Figure 2 shows the reconstruction performance of all three models on random projections. In all tasks the RIM clearly outperforms both other models, showing overall consistent convergence behavior. The FFN performs well on easier tasks but starts to show degrading performance over time on more difficult tasks. This suggests that the state information of RIM plays an important role on the convergence behavior as well as overall performance. The GDN shows worst performance among all three models. For all tasks, the performance of GDN starts to degrade clearly after the 20 time steps that were used during training. We hypothesize that the model is able to compensate some of the missing information about the current estimate of $\mathbf{x}$ through state variable $\mathbf{s}$ during training, but the model is not able to transfer this ability to episodes with more iterations.

These results suggests that both the current estimate as well as the recurrent state carry useful information for performing inference. We will therefor only consider fully fledged RIMs from here on.

4.5 Image Denoising

After evaluating our model on 32 x 32 pixel image patches we wanted to see how reconstruction performance generalizes to full sized images and to an out of domain problem. We chose to reuse the RIM that was trained on the random projections task to perform image denoising. In this section we will call this model RIM-3task. To test the hypothesis that inference should be trained task specific, we further trained a model RIM-denoise solely on the denoising task. Table 2 shows the denoising performance through the mean PSNR on the BSD-300 test set for both models as compared to state-of-the-art methods in image denoising. The RIM-3task model shows very competitive results with other methods on all noise levels. This exemplifies that the model indeed has learned something reminiscent of a prior, as it was never directly trained on this task. The RIM-denoise model further improves upon the performance of RIM-3task and it outperforms most other methods on all noise levels. This is to say that the same RIM was used to perform denoising on different noise levels, and this model does not require any hand tuning after training.

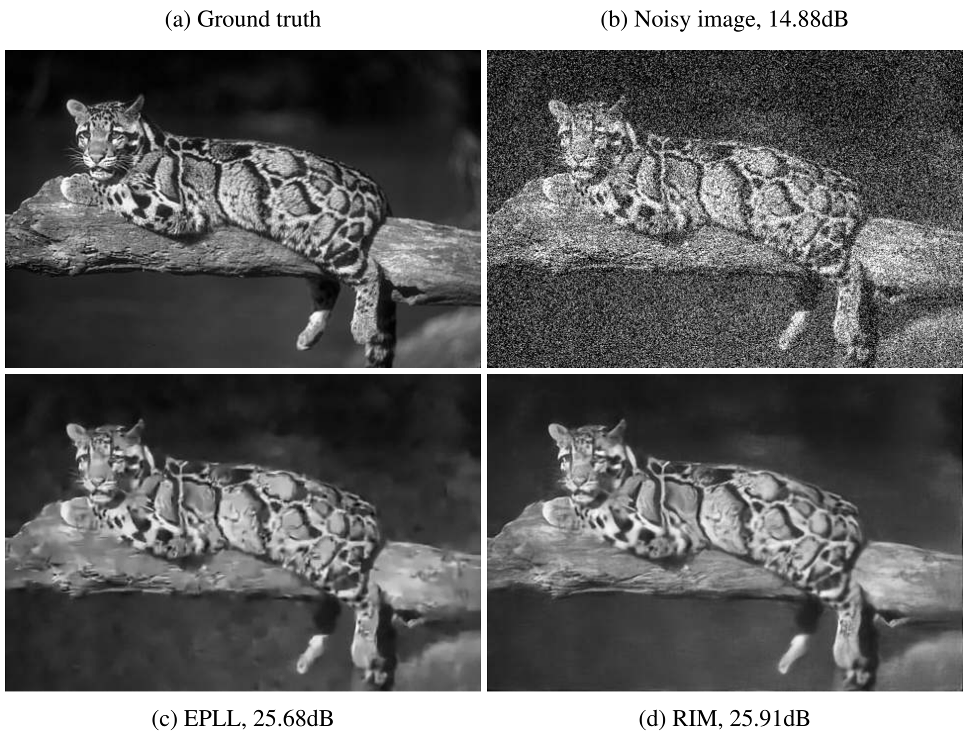

Figure 3: *Denoising performance on example image use in Zoran & Weiss (2011).* $\sigma = 50$. Noisy image was 8-bit quantized before reconstruction.

Table 2 shows denoising perfomance on image that have been 8-bit quantized after adding noise(see Schmidt et al. (2016)). In this case performance slightly deteriorates for both models, though still making competitive with state-of-the-art methods. This effect could possibly be accommodated through further training, or by adjusting the forward model. Figure 3 gives some qualitative results on the denoising performance for one of the test images from BSD-300 as compared to the method from Zoran & Weiss (2011). RIM is able to produce more naturalistic images with less visible artifacts. The state variable in our RIM model allows for a growing receptive field size over time, which could explain the good long range interactions that the model shows.

Method	PSNR
CBM3D	30.18
RTF-5	30.57
RIM (ours)	$\mathbf{30.84}$(30.67)

Table 1: *Color denoising.* Denoising performance on the 68 images for $\sigma = 25$ after 8-bit quantization. Results for RTF-5 (Schmidt et al., 2016) and CBM3D (Dabov et al., 2007b) adopted from Schmidt et al. (2016). In parenthesis are results for the full 100 test images.

Many denoising algorithms are solely tested on gray-scale images. Sometimes this is due to additional difficulties that multi-channel problems bring for some inference approaches. To show that it is straightforward to apply RIMs to multi-channel problems we trained a model to denoise RGB images. The denoising performance can be seen in table 1. The model is able to exploit correlations across color channels which allows for an additional boost in reconstruction performance.

4.6 Image Super-resolution

We further tested our approach on the well known image super-resolution task. We trained a single RIM [5] on 36 x 36 pixel image patches from the BSD-300 training set to perform image super-

[5]The architecture of this model was slightly simplified in comparison to the previous problems. Instead of strided convolutions, we chose *a trous* convolutions. This model is more flexible and used only about 500.000 parameters. Previous experiments will be updated with the same model architecture.

	Not Quantized			8-bit Quantized		
σ	15	25	50	15	25	50
KSVD	30.87	28.28	25.17			
5x5 FoE	30.99	28.40	25.35		28.22	
BM3D	31.08	28.56(28.35)	25.62(25.45)		28.31	
LSSC	31.27	28.70	25.72		28.23	
EPLL	31.19	28.68(28.47)	25.67(25.50)			
opt-MRF	31.18	28.66	25.70			
MLP		28.85(28.75)	(25.83)			
RTF-5		28.75			28.74	
RIM-3task	31.19(30.98)	28.67(28.45)	25.78(25.59)	31.06(30.88)	28.41(28.24)	24.86(24.73)
RIM-denoise	**31.31**(31.10)	**28.91**(28.72)	**26.06**(25.88)	**31.25**(31.05)	**28.76**(28.58)	**25.27**(25.14)

Table 2: *Denoising performance on gray-scale images from BSD-300 test set.* Shown are mean PSNR values for different noise values. Number outside of parenthesis correspond to test performance on the 68 test images from Roth & Black (2005), and numbers in parenthesis correspond to performance on all 100 test images from BSD-300. 68 image performance for KSVD (Elad & Aharon, 2006), FoE (Roth & Black, 2005), BM3D (Dabov et al., 2007a), LSSC (Mairal et al., 2009), EPLL (Zoran & Weiss, 2011), and opt-MRF (Chen et al., 2013) adopted from Chen et al. (2013). Performances on 100 images adopted from Burger et al. (2013). 68 image performance on MLP (Burger et al., 2012), RTF-5 (Schmidt et al., 2016) and all quantized results adopted from Schmidt et al. (2016).

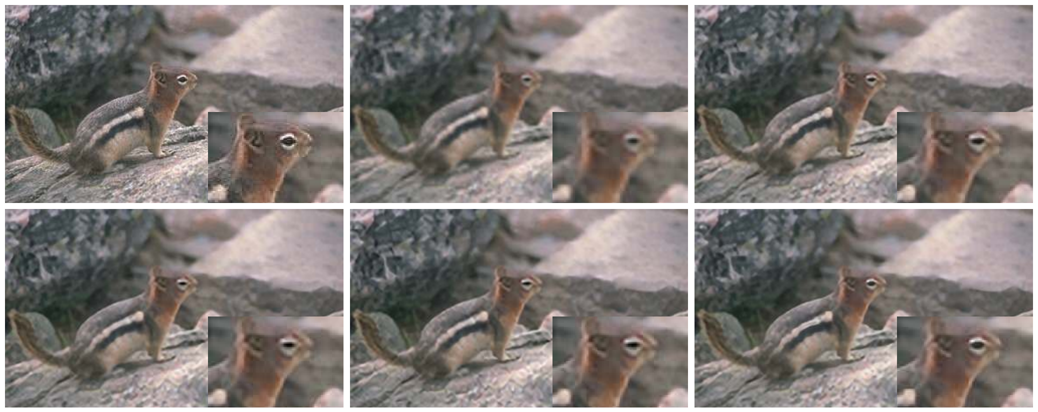
(a) Original Image
(b) Bicubic: 30.43/0.8326
(c) SRCNN: 31.34/0.8660
(d) A+: 31.43/0.8676
(e) SelfExSR: 31.18/0.8656
(f) RIM: **31.59/0.8712**

Figure 4: *Super-resolution example with factor 3.* Comparison with the same methods as in table 3. Reported numbers are PSNR/SSIM. Best results in bold.

resolution for factors 2, 3, and 4[6]. We followed the same testing protocol as in Huang et al. (2015), and we used the test images that were retrieved from their website [7]. Table 3 shows a comparison with some state-of-the-art methods on super-resolution for the BSD-300 test set. Figure 4 shows a qualitative example of super-resolution performance. The other deep learning method in this comparison, SRCNN Dong et al. (2014), is outperformed by RIM on all scales. Interestingly SRCNN was trained for each scale independently whereas we only trained one RIM for all scales. The chosen RIM has only about 500.000 parameters which amounts to about 2MB of disk space, which makes this architecture very attractive also for mobile computing.

[6]We reimplemented MATLABs bicubic interpolation kernel in order to apply a forward model (subsampling) in TensorFlow which agrees with the forward model in Huang et al. (2015).

[7]https://sites.google.com/site/jbhuang0604/publications/struct_sr

Metric	Scale	Bicubic	SRCNN	A+	SelfExSR	**RIM (Ours)**
PSNR	2x	29.55 ± 0.35	31.11 ± 0.39	31.22 ± 0.40	31.18 ± 0.39	$\mathbf{31.39} \pm 0.39$
	3x	27.20 ± 0.33	28.20 ± 0.36	28.30 ± 0.37	28.30 ± 0.37	$\mathbf{28.51} \pm 0.37$
	4x	25.96 ± 0.33	26.70 ± 0.34	26.82 ± 0.35	26.85 ± 0.36	$\mathbf{27.01} \pm 0.35$
SSIM	2x	0.8425 ± 0.0078	0.8835 ± 0.0062	0.8862 ± 0.0063	0.8855 ± 0.0064	$\mathbf{0.8885} \pm 0.0062$
	3x	0.7382 ± 0.0114	0.7794 ± 0.0102	0.7836 ± 0.0104	0.7843 ± 0.0104	$\mathbf{0.7888} \pm 0.0101$
	4x	0.6672 ± 0.0131	0.7018 ± 0.0125	0.7089 ± 0.0125	0.7108 ± 0.0124	$\mathbf{0.7156} \pm 0.0125$

Table 3: *Image super-resolution performance on RGB images from BSD-300 test set.* Mean and standard deviation (of the mean) of Peak Signal-to-Noise Ratio (PSNR) and Structural Similarity Index (SSIM) Wan (2004). Standard deviation of the mean was estimated from 10.000 boostrap samples. Test protocol and images taken from Huang et al. (2015). Only the three best performing methods from Huang et al. (2015) were chosen for comparison: SRCNN Dong et al. (2014), A+ Timofte et al. (2015), SelfExSR Huang et al. (2015). Best mean values in bold.

5 Discussion

In this work, we introduce a general learning framework for solving inverse problems with deep learning approaches. We establish this framework by abandoning the traditional separation between model and inference. Instead, we propose to learn both components jointly without the need to define their explicit functional form. This paradigm shift enables us to bridge the gap between the fields of deep learning and inverse problems. We believe that this framework can have a major impact on many inverse problems, for example in medical imaging and radio astronomy. Although we have focused on linear image reconstruction tasks in this work, the framework can be applied to inverse problems of all kinds, such as non-linear inverse problems.

Acknowledgments

The research was funded by the DOME project (Astron & IBM) and the Netherlands Organization for Scientific Research (NWO). The authors are greatful for helpful comments from Thomas Kipf, Mijung Park, Rajat Thomas, and Karen Ullrich.

References

Image quality assessment: form error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, 2004.

Michal Aharon, Michael Elad, and Alfred Bruckstein. K-SVD: An algorithm for designing overcomplete dictionaries for sparse representation. *IEEE Transactions on Signal Processing*, 54(11): 4311–4322, nov 2006.

Marcin Andrychowicz, Misha Denil, Sergio Gomez, Matthew W. Hoffman, David Pfau, Tom Schaul, and Nando de Freitas. Learning to learn by gradient descent by gradient descent. jun 2016.

Harold Christopher Burger, Christian Schuler, and Stefan Harmeling. Image denoising: Can plain neural networks compete with BM3D? In *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2392–2399. IEEE, jun 2012.

Harold Christopher Burger, Christian J. Schuler, and Stefan Harmeling. Learning how to combine internal and external denoising methods. In Joachim Weickert, Matthias Hein, and Bernt Schiele (eds.), *GCPR*, volume 8142 of *Lecture Notes in Computer Science*, pp. 121–130. Springer, 2013.

Emmanuel J. Candès, Justin K. Romberg, and Terence Tao. Stable signal recovery from incomplete and inaccurate measurements. *Communications on Pure and Applied Mathematics*, 59(8):1207–1223, aug 2006.

Yunjin Chen, Thomas Pock, René Ranftl, and Horst Bischof. Revisiting Loss-Specific Training of Filter-Based MRFs for Image Restoration. In *35th German Conference on Pattern Recognition (GCPR)*, pp. 271–281, 2013.

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. dec 2014.

K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian. Image Denoising by Sparse 3-D Transform-Domain Collaborative Filtering. *IEEE Transactions on Image Processing*, 16(8):2080–2095, aug 2007a.

Kostadin Dabov, Alessandro Foi, Vladimir Katkovnik, and Karen Egiazarian. Color Image Denoising via Sparse 3D Collaborative Filtering with Grouping Constraint in Luminance-Chrominance Space. In *2007 IEEE International Conference on Image Processing*, pp. I – 313–I – 316. IEEE, sep 2007b.

Chao Dong, Chen Change Loy, Kaiming He, and Xiaoou Tang. Learning a deep convolutional network for image super-resolution. *ECCV*, pp. 184–199, 2014.

David L. Donoho. For most large underdetermined systems of linear equations the minimal L1-norm solution is also the sparsest solution. *Communications on Pure and Applied Mathematics*, 59(6): 797–829, jun 2006a.

D.L. Donoho. Compressed sensing. *IEEE Transactions on Information Theory*, 52(4):1289–1306, apr 2006b.

Michael Elad and Michal Aharon. Image Denoising Via Sparse and Redundant Representations Over Learned Dictionaries. *IEEE Transactions on Image Processing*, 15(12):3736–3745, dec 2006.

Mário A. T. Figueiredo, Robert D. Nowak, and Stephen J. Wright. Gradient Projection for Sparse Reconstruction: Application to Compressed Sensing and Other Inverse Problems. *IEEE Journal of Selected Topics in Signal Processing*, 1(4):586–597, dec 2007.

Karol Gregor and Yann LeCun. Learning Fast Approximations of Sparse Coding. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pp. 399–406, 2010.

Karol Gregor, Frederic Besse, Danilo Jimenez Rezende, Ivo Danihelka, and Daan Wierstra. Towards Conceptual Compression. apr 2016.

Jia-Bin Huang, Abhishek Singh, and Narendra Ahuja. Single image super-resolution from transformed self-exemplars. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5197–5206. IEEE, jun 2015.

Diederik P. Kingma and Max Welling. Auto-Encoding Variational Bayes. In *The 2nd International Conference on Learning Representations (ICLR)*, 2014.

Sanjiv Kumar, Jonas August, and Martial Hebert. Exploiting Inference for Approximate Parameter Learning in Discriminative Fields: An Empirical Study. In *Proceedings of the 5th international conference on Energy Minimization Methods in Computer Vision and Pattern Recognition*, pp. 153–168. Springer-Verlag, 2005.

Julien Mairal, Francis Bach, Jean Ponce, Guillermo Sapiro, and Andrew Zisserman. Non-local sparse models for image restoration. In *2009 IEEE 12th International Conference on Computer Vision*, pp. 2272–2279. IEEE, sep 2009.

David Martin, Charless Fowlkes, Doron Tal, and Jitendra Malik. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *Proc. 8th Int'l Conf. Computer Vision*, volume 2, pp. 416–423, July 2001.

Hannes Nickisch and Matthias W. Seeger. Convex variational Bayesian inference for large scale generalized linear models. In *Proceedings of the 26th International Conference on Machine Learning*, pp. 761–768, New York, New York, USA, jun 2009. ACM Press.

D J Rezende, S Mohamed, and D Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *Proceedings of The 31st International Conference on Machine Learning*, pp. 1278–1286, 2014.

Stefan Roth and Michael J. Black. Fields of experts: A framework for learning image priors. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 2, pp. 860–867. IEEE, 2005.

Uwe Schmidt, Jeremy Jancsary, Sebastian Nowozin, Stefan Roth, and Carsten Rother. Cascades of regression tree fields for image restoration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(4):677–689, 2016.

Radu Timofte, Vincent de Smet, and Luc van Gool. A+: Adjusted anchored neighborhood regression for fast super-resolution. In *ACCV*, volume 9006, pp. 111–126. 2015.

Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning*, pp. 1096–1103, New York, New York, USA, 2008. ACM Press.

Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, and Pierre-Antoine Manzagol. Stacked Denoising Autoencoders: Learning Useful Representations in a Deep Network with a Local Denoising Criterion. *The Journal of Machine Learning Research*, 11:3371–3408, 2010.

MJ Wainwright. Estimating the wrong graphical model: Benefits in the computation-limited setting. *The Journal of Machine Learning Research*, 2006.

Daniel Zoran and Yair Weiss. From learning models of natural image patches to whole image restoration. In *2011 International Conference on Computer Vision*, pp. 479–486. IEEE, nov 2011.