Practical Techniques For Using Neural Networks To Estimate State From Images

Stephen C. Ashmore
Department of Computer Science
and Computer Engineering
University of Arkansas
Fayetteville, Arkansas 72701, USA
scashmor@uark.edu

Michael S. Gashler
Department of Computer Science
and Computer Engineering
University of Arkansas
Fayetteville, Arkansas 72701, USA
mgashler@uark.edu

***Abstract*—An important task for training a robot (virtual or real) is to estimate state. State includes the state of the robot and its environment. Images from digital cameras are commonly used to monitor the robot due to the rich information, and low-cost hardware. Neural networks excel at catagorizing images, and should prove powerful to estimate the state of the robot from these images. There are many problems that occur when attempting to estimate state with neural networks, including high resolution of images, training time, vanishing gradient, and more. This paper presents several practical techniques for facilitating state estimation from images with neural networks.**

I. Introduction

An important task in training a robot controller with machine learning techniques is estimating the state of the robot and the environment in which it operates. Digital cameras have increasingly become a very compelling option for monitoring robots. Due to the rich amount of information that visual images can provide, the ease with which humans can relate to them, and the proliferation of low-cost digital camera hardware, they have largely eclipsed other methods for monitoring state. However, estimating state from digital images can be a daunting task. This paper aggregates a collection of practical techniques that the authors have found to be useful for facilitating state estimation from images with neural networks.

Recently, deep neural network learning techniques have made significant progress in working with images, excelling at such tasks as image recognition, segmentation, and localization [1]. In particular, convolutional neural networks have largely claimed the spotlight due to their proficiency in image classification tasks [2], [3], [4]. These advances have led to much speculation that highly proficient robots would soon be forthcoming [5], [6], [7], [8]. However, classifying images is quite a different task from estimating the state they represent. There remains somewhat of a gap in the current literature between the many well-established techniques for image recognition and the practical knowledge necessary to apply deep learning methods for estimating state from digital images. This paper fills some of that gap by describing a collection of methods that have been found to be effective for this purpose.

II. Parameterize the pixels in target images

Autoencoders are a popular method for estimating state from images by reducing them to a low-dimensional representation [9]. They are neural networks that attempt to approximate the identity function, as illustrated in Figure 1. Traditional autoencoders pass their values through a bottleneck layer with only a few nodes. The activations in this small layer form a low-dimensional encoding of the inputs. The portion of the neural network that computes this encoding is referred to as the "encoder", and the remainder is the "decoder".

Since convolutional neural networks have recently performed so well at the task of digesting images, it is natural to use convolutional layers in the encoder [10]. But what is the best way to handle large images in the decoder, on the output end of the autoencoder? A surprisingly simple approach is to make the decoder predict only the values of a single pixel [11]. Two additional inputs are added to the decoder to specify which pixel should be predicted. This modification adds several desirable properties to the model:

- Images can be generated with arbitrary resolution because they are represented as a 2-D function, instead of a 2-D array of pixels.
- During training, it is not necessary to sample every pixel of every image. Using a random sampling of pixels is generally more efficient, since each pixels usually has a color similar to its neighbors.
- The number of weights in the decoder is no longer tied to the resolution of the images. Typically, the number of weights is much smaller, which also improves training time.

Interestingly, this modification makes the decoder complement an encoder that uses convolutional layers. Whereas convolutional layers apply the same kernel weights at each position in the input image to produce the values it feeds into the next layer, a decoder that predicts the values of only a single parameterized pixel applies the same neural network to compute the values of each pixel in the output image. And, whereas convolutional layers greatly reduce the number of weights necessary to digest input images, this approach greatly reduces the number of weights necessary to generate

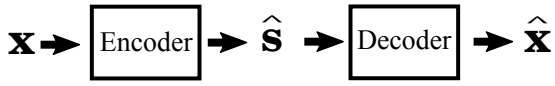

Fig. 1. A diagram of a traditional autoencoder.

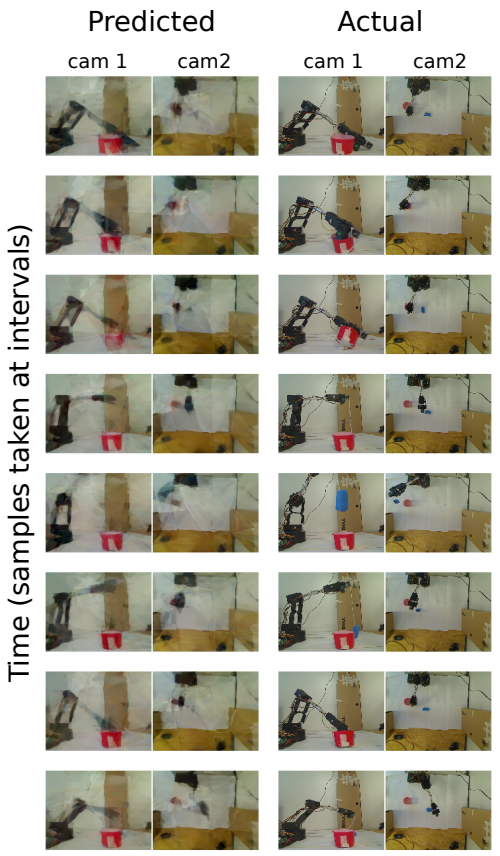

Fig. 2. **Left:** Anticipated observations computed from an internal representation of state. **Right:** Actual images from digital cameras positioned to observe a low-cost robotic arm.

output images. Example results obtained using this technique are shown in Figure 2.

III. Move High-Dim Inputs to the Output End

A common false assumption with neural networks is that "inputs" must always be fed into the input end of the network. However, because neural networks are differentiable, gradient descent can also be used to infer values that could be fed into the input end of the network to yield particular values on the output end [12]. The *update rule* for the inputs is derived from the same backpropagation algorithm used to compute the update rule for the weights. The blame terms that backpropagation computes for the units of the input layer are simply multiplied by the weights that feed into that layer to obtain the gradient for the inputs. By adjusting the inputs in the direction of this gradient, the network predicts values closer to the target values. So, latent input values can be initialized to some arbitrary values, such as the origin, then updated by gradient descent until the network predictions match the target values.

The cost of attaching sensors to the output end of the network is that it takes more than a single feed-forward pass to infer values on the other end. Although this cost can be significant, there are also some benefits that often outweigh it:

- The weights near the output end of a neural network train faster than weights near the input end.

function estimate_state_from_images($\mathbf{X}$)

Assume $\mathbf{X}$ is a training set containing n images.
Let $\mathbf{S}$ be a set of n latent state vectors, initialized to $\mathbf{0}$.
until convergence is detected, **do**:
  for each $\mathbf{x}_i \in \mathbf{X}$ in random order:
    for a number of random pixels drawn from $\mathbf{x}_i$:
      Feed in the pixel coordinates and $\mathbf{s}_i$.
      Use backpropagation to update weights and $\mathbf{s}_i$.
return $\mathbf{S}$.

Fig. 3. Pseudo-code to train a stand-alone decoder and simultaneously estimate the state corresponding with each training image.

- Neural networks do not generally tolerate missing or sparse input values. However, missing outputs are trivially handled by using no error (that is, assuming the missing target values match the predictions).
- To handle very sparse values, the computation associated with the missing elements can be omitted entirely.

IV. Train a Stand-Alone Decoder

Instead of using a full autoencoder to estimate state, an alternative design is to use only a decoder and infer the values of the encoding. This approach has been demonstrated to be highly effective for imputing missing values in data [13], [14]. With this design, the decoder is used in a bidirectional manner, as described with pseudocode in Figure 3.

Two important advantages are obtained from this technique:

- The encoder was previously the only remaining constraint on the resolution of the observed images. By removing the encoder as a dependency, now cameras with arbitrary resolution may be used.
- The encoder implicitly applied a smoothing constraint to the representations of state. That is a good thing when observations are low in dimensionality. With images, however, better results are often obtained without it.

To illustrate this last advantage, an autoencoder and stand-alone decoder were both trained on the MNIST dataset of handwritten digits. The autoencoder used a topology of $784 \rightarrow 200 \rightarrow 50 \rightarrow 10 \rightarrow 50 \rightarrow 200 \rightarrow 784$. The stand-alone decoder used a topology of $10 \rightarrow 50 \rightarrow 200 \rightarrow 784$. The standard split of 60,000 samples was used for training, and 10,000 samples for testing. (For the stand-alone decoder, inference was used to estimate state for the test samples as well, but the weights of the decoder were not updated when test images were presented.) Figure 4 shows root mean squared error for reconstructing the images over wall-clock training time. The autoencoder trained much faster than the inference approach. It achieved its best results after only 20 seconds of training, then began to overfit the training data. The inference approach surpassed the autoencoder after 170 seconds of training, and was still improving after 600 seconds.

Notably, the inference approach does not always beat the autoencoder approach. When only smaller subsets of the MNIST training data were used for training, for example, the autoencoder approach did better. The effectiveness of the inference approach seems to depend on having an abundance of uncorrelated training data to guide the inference of state.

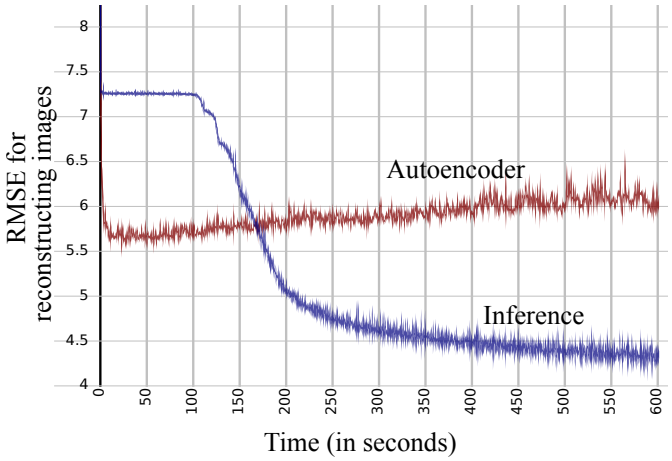

Fig. 4. Inference with a stand-alone decoder beats an autoencoder for reconstructing images from the MNIST dataset.

V. Passively train an encoder

The iterative process necessary to infer state using a stand-alone decoder may seem cumbersome in comparison with an encoder, since using an encoder would require only a single feed-forward pass to estimate state. However, with the stand-alone decoder, it should be noted that an accurate estimate of state may be inferred without visiting every pixel in the observed image. When observed images are sufficiently high in resolution, the stand-alone decoder becomes the faster approach.

The advantages of both techniques may be leverage by passively training an encoder on scaled-down versions of the observed images. This is done by using the scaled-down images as features or inputs for training the encoder, and the latent estimates of state as labels or target values. This approach is *passive* in that the training of the encoder does not influence the decoder in any way. Rather, it depends on the decoder, which is the opposite of how autoencoders behave. Training an encoder in this passive manner brings the following desirable properties:

- Training time is reduced because the encoder follows the decoder, rather than leads it. By contrast, in an autoencoder, the encoder is the farthest part from the output units. Thus, due to the problem of vanishing gradients [15], the encoder responds most slowly to training patterns. Therefore, removing it as a dependency of training improves overall training time.
- After training, the encoder can be used to accelerate inference of state for novel (out-of-band) observations. This is done by scaling down the new observation and feeding it through the encoder to compute an initial estimate of state. The state is then further refined using inference iterations from pixels sampled from the full resolution images. Having a reasonable initial estimate of state significantly reduces the number of inference iterations needed.
- Since the scaled-down images are used only with the encoder, and the decoder still uses full resolution images, the resolution of the scaled-down images may be tuned without regard to its impact on the final precision of state estimates.

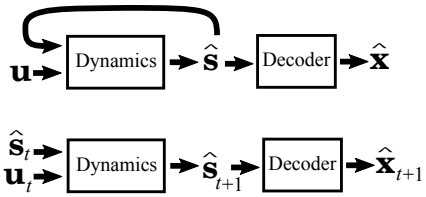

Fig. 5. **Top:** A recurrent neural network sufficient to model the dynamics of a robot. **Bottom:** State estimates for a training sequence of observations can be used to train recurrent models with simple supervised learning methods.

VI. Nonlinear dimensionality reduction excels at pretraining

Nonlinear dimensionality reduction (NLDR) methods are designed to compute low-dimensional representations of state from high-dimensional observations. It follows that instead of initializing the states with zeros, NLDR methods could be used to pretrain the initial estimates of state (before iterations of gradient descent are applied to train the decoder and further refine the representations of state). However, NLDR methods typically have two properties that make them ill-suited for this task: (1) They require a necessary neighbor-finding step that is prone to introduce undesired connections between irrelevant states [16], and (2) they typically assume that all observations are independent of each other.

These problems can be mitigated by utilizing the small-but-significant additional information found in the knowledge that robot observations are not independent of each other, but are made in a sequence. This implicitly gives most observations two known neighbors: the one that precedes it in the sequence, and the one that follows it. By computing the distance between these neighbors, a local radius can be determined that is suitable for finding other observations that should probably be considered to be neighbors, without attaching to states that should not be connected.

VII. Pretrain states in recurrent models of dynamics

Another important challenge in applying machine learning techniques to robots is modeling dynamics. Models of dynamics predict how the state will change when given control vectors are applied to the robot. Recurrent neural networks are typically used to model dynamics because each new state depends on the previous state, as well as the control vector that is applied to the robot. The most common method for training recurrent neural networks is backpropagation through time (BPTT) [17], [18]. Unfortunately, recurrent models have developed a reputation for being difficult to manage because BPTT is cumbersome to implement, converges very slowly, and is highly susceptible to local optima [19], [20], [21].

When state has already been estimated for each observation in the training data, a much easier solution becomes possible: Simply use both the *before* and *after* estimates of state in each pattern presentation for training the model of dynamics. With this approach, regular supervised learning approaches, such as stochastic gradient descent, become effective for training

recurrent models, as illustrated in Figure 5. The recurrent connections become irrelevant during such training because the state estimates already capture the transitions within state. Since digital images often contain such rich information about state, this approach can sometimes even beat BPTT [11].

It should be noted that many control models, such as deep Q-networks, typically feed observations as inputs into the model [22]. This is done so that the model can estimate state from the observations. However, as described previously in the context of autoencoders, state may be estimated more effectively in many cases by inference techniques when observations are moved to the output end of the model. Therefore, these techniques can still be combined with model-based reinforcement learning and other control models.

VIII. Keep the state rational

A significant challenge with learned models of dynamics is that they are inevitably imperfect. When a predicted state transition differs from the actual state by a small amount, this error will feed back into the model to influence the next estimate of state. When errors feed back into the system, they can compound quickly, leading to irrational predictions. However, several simple techniques can be applied to mitigate this effect:

- When new observations are made, a few iterations of gradient descent can be used to refine the sate. This has a strong effect of keeping the state accurate. Unfortunately, this approach cannot be used during open-loop planning, when no new observations are being made.
- Using as few dimensions as possible for the internal representation of state will help constrain the system to stay in rational states.
- Contractive regularization [23] can help to influence the model to use as much of the state space as possible for representations of rational states. When a smaller portion of the state space is used to represent irrational states, there is lower likelihood of the state drifting into one of these regions, where predictions will decay quickly.
- If the training observations are believed to thoroughly sample the space of rational states, another possibility is to add a soft constraint that weakly pulls the current representation of state to within some radius of the nearest state computed to correspond with one of the training observations. Or, another model could be trained to estimate the rationality of a given state, and used to push the state away from irrational states.

IX. Conclusion

Estimating the state represented within images is an important but challenging task at the intersection of machine learning and robotics. This paper presented a collection of several techniques that its authors have found to be effective for this task. Many of them may be considered to be simple and/or obvious, but it is unlikely that all of them are known to the community, so this paper seeks to provide a useful resource for practitioners who use machine learning with robots.

References

[1] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural Networks*, vol. 61, pp. 85–117, 2015.

[2] H. Lee, R. Grosse, R. Ranganath, and A. Y. Ng, "Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations," in *Proceedings of the 26th Annual International Conference on Machine Learning*. ACM, 2009, pp. 609–616.

[3] D. Cireşan, U. Meier, J. Masci, and J. Schmidhuber, "A committee of neural networks for traffic sign classification," in *Neural Networks (IJCNN), The 2011 International Joint Conference on*. IEEE, 2011, pp. 1918–1921.

[4] A. Krizhevsky, I. Sutskever, and G. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems 25*, 2012, pp. 1106–1114.

[5] G. A. Pratt, "Is a cambrian explosion coming for robotics?" *The Journal of Economic Perspectives*, vol. 29, no. 3, pp. 51–60, 2015.

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

[7] D. Harris, "Robots helped inspire deep learning and might become its killer app," 2014. [Online]. Available: https://gigaom.com/2014/07/29/robots-helped-inspire-deep-learning-and-might-become-its-killer-app/

[8] C. Metz, "'deep learning will soon give us super-smart robots," 2015. [Online]. Available: http://www.wired.com/2015/05/remaking-google-facebook-deep-learning-tackles-robotics/

[9] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 28, no. 313 (5786), pp. 504–507, July 2006.

[10] J. Masci, U. Meier, D. Cireşan, and J. Schmidhuber, "Stacked convolutional auto-encoders for hierarchical feature extraction," in *International Conference on Artificial Neural Networks*. Springer, 2011, pp. 52–59.

[11] M. S. Gashler and T. R. Martinez, "Temporal nonlinear dimensionality reduction," in *Proceedings of the International Joint Conference on Neural Networks*. IEEE Press, 2011, pp. 1959–1966.

[12] G. E. Hinton, "Generative back-propagation," *Abstracts 1st INNS*, 1988.

[13] M. Scholz, F. Kaplan, C. L. Guy, J. Kopka, and J. Selbig, "Non-linear PCA: a missing data approach," *Bioinformatics*, vol. 21, no. 20, pp. 3887–3895, 2005.

[14] M. S. Gashler, M. R. Smith, R. Morris, and T. Martinez, "Missing value imputation with unsupervised backpropagation," *Computational Intelligence*, 2014. [Online]. Available: http://arxiv.org/abs/1312.5394

[15] S. Hochreiter, "The vanishing gradient problem during learning recurrent neural nets and problem solutions," *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, vol. 6, no. 02, pp. 107–116, 1998.

[16] M. S. Gashler and T. R. Martinez, "Robust manifold learning with CycleCut," *Connection Science*, vol. 24, no. 1, pp. 57–69, 2012. [Online]. Available: http://www.tandfonline.com/doi/abs/10.1080/09540091.2012.664122

[17] P. J. Werbos, "Generalization of backpropagation with application to a recurrent gas market model," *Neural Networks*, vol. 1, no. 4, pp. 339–356, 1988.

[18] M. C. Mozer, "A focused backpropagation algorithm for temporal pattern recognition," *Backpropagation: theory, architectures, and applications*, pp. 137–169, 1995.

[19] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE transactions on neural networks*, vol. 5, no. 2, pp. 157–166, 1994.

[20] R. Pascanu, T. Mikolov, and Y. Bengio, "On the difficulty of training recurrent neural networks." *ICML (3)*, vol. 28, pp. 1310–1318, 2013.

[21] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Advances in neural information processing systems*, 2014, pp. 3104–3112.

[22] M. Hausknecht and P. Stone, "Deep recurrent q-learning for partially observable mdps," *arXiv preprint arXiv:1507.06527*, 2015.

[23] S. Rifai, P. Vincent, X. Muller, X. Glorot, and Y. Bengio, "Contractive auto-encoders: Explicit invariance during feature extraction," in *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, 2011, pp. 833–840.