Feedback Networks

Amir R. Zamir[1,2*] Te-Lin Wu[1*] Lin Sun[1] William Shen[1] Jitendra Malik[2] Silvio Savarese[1]
[1] Stanford University [2] University of California, Berkeley
http://feedbacknet.stanford.edu/

Abstract

Currently, the most successful learning models in computer vision are based on learning successive representations followed by a decision layer. This is usually actualized through feedforward multilayer neural networks, e.g. ConvNets, where each layer forms one of such successive representations. However, an alternative that can achieve the same goal is a feedback based approach in which the representation is formed in an iterative manner based on a feedback received from previous iteration's output.

We establish that a feedback based approach has several fundamental advantages over feedforward: it enables making early predictions at the query time, its output naturally conforms to a hierarchical structure in the label space (e.g. a taxonomy), and it provides a new basis for Curriculum Learning. We observe that feedback networks develop a considerably different representation compared to feedforward counterparts, in line with the aforementioned advantages. We put forth a general feedback based learning architecture with the endpoint results on par or better than existing feedforward networks with the addition of the above advantages. We also investigate several mechanisms in feedback architectures (e.g. skip connections in time) and design choices (e.g. feedback length). We hope this study offers new perspectives in quest for more natural and practical learning models.

1. Introduction

Feedback is defined to occur when (full or partial) outputs of a system are routed back into the system as part of an iterative cause-and-effect process [12]. Utilizing feedback is a strong way of making predictions in various fields, ranging from control theory to psychology [28, 37, 2]. Employing feedback connections is also heavily exercised by biological organisms and the brain [17, 38, 38, 7, 29] suggesting a core role for it in complex cognition. In this paper, we establish that a feedback based learning approach has several core advantages over the commonly employed feedforward paradigm making it a worthwhile alternative. These advantages (elaborated below) are mostly attributed to the fact that the final prediction is made in an iterative, rather than one-time, manner along with an explicit notion of the thus-far output at each iteration.

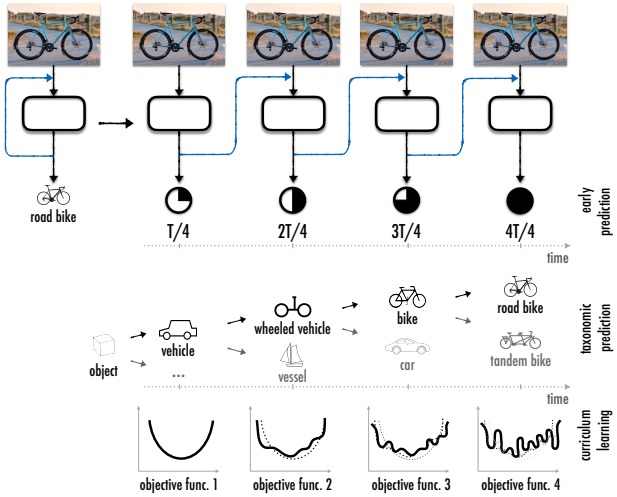

Figure 1. **A feedback based learning model.** The basic idea is to make predictions in an iterative manner based on a notion of the thus-far outcome. This provides several core advantages: I. enabling early predictions (given total inference time T, early predictions are made in fractions of T); II. naturally conforming to a taxonomy in the output space (one branch of the taxonomy is explored with each iteration); and III. better grounds for curriculum learning.

Early Predictions: One advantage is providing early predictions, meaning that estimations of the output can be provided in a fraction of total inference time. This is schematically illustrated in Fig. 1. This property is a result of iterative inference and is in contrast to feedforward where a one-time output is provided only when the signal reaches the end of the network. This is of particular importance in practical scenarios, such as robotics or autonomous driving; e.g. imagine a self driving car that receives a cautionary heads up about possibly approaching a pedestrian on a highway, without needing to wait for the final definite out-

*Both authors contributed equally.

put. Such scenarios are abundant in practice as usually time is crucial and limited computation resources can be reallocated based on early predictions on-the-fly, given a proper uncertainty measure, such as Minimum Bayes Risk [27].

Taxnomoy Compliance: Another advantage is making predictions that naturally conform to a hierarchical structure in the output space, e.g. a taxonomy, even when not trained using the taxonomy. The early predictions of the feedback model conform to a coarse classification, while the later iterations further decompose the coarse class into finer classes. This is illustrated in Fig.1. This is again due to the fact that the predictions happen in an iterative manner coupled with a *coarse-to-fine representation*. The coarse-to-fine representation is naturally developed as a results of the feedback based training, as the network is forced to make a prediction about the output as early as the first iteration and in all following iterations.

Episodic Curriculum Learning: The previous advantage is closely related to the concept of Curriculum Learning [4], where gradually increasing the complexity of the learning task leads to training a better learner (as compared to no ordering based on complexity). Curriculum Learning is an important mechanism employed both in machine learning systems as well as human education [11, 4, 26]. For non-convex training criteria (such as in ConvNets), a curriculum is known to assist with finding better minima; in convex cases, curriculum improves the convergence speed [4]. This is due to smoothing the objective criteria through the curriculum to guide the optimization towards better minima.

Since prediction in a feedforward network happens in a one-time manner, the only opportunity for enforcing a curriculum is through presenting the training data to the same full network in an order based on complexity (i.e. first epochs formed of easy examples, and later the hard ones). In contrast, the predictions in a feedback based model happen in an iterative form, and this enables enforcing a curriculum *through the episodes of prediction for one query*, which we call *Episodic Curriculum Learning*. In other words, sequential easy-to-hard predictions can be enforced to be made for one datapoint (e.g., training the early episodes to predict the type of animal and the later episodes to predict the particular breed) without the need for any ordering in training datapoints. Therefore, any taxonomy can be immediately used as a curriculum strategy.

In our model, we define feedback based prediction as a recurrent (weight) shared operation, where at each iteration, the output is estimated and passed onto the next iteration through a hidden state. The next iteration then makes an updated prediction using the shared operation and received hidden state. It is crucial for the hidden state to carry a direction notion of output, otherwise the entire system would be a feedforward pass realized through a recurrent operation [31]. Therefore, we train the network to make a prediction at each iteration by backpropagating the loss in all iteration. We put forth a generic working architecture for such networks and empirically prove the aforementioned advantages on various datasets. Though we show that the feedback approach achieves competent final results, the primary goal of this paper is to establish the aforementioned conceptual properties, rather than optimizing for endpoint performance on any benchmark. The developed architectures, pretrained models, and demos reproducing the reported numbers are available at `http://feedbacknet.stanford.edu/`.

2. Related Work

Conventional feedforward networks, such as AlexNet [25] or VGG [39], do not employ either recurrence or feedback like mechanisms. Recently, several successful efforts introduced recurrence-inspired mechanisms in feedforward models. ResNet [14] is one nominal example introducing parallel residual connections, as well as hypernetworks [13], highway networks [42], stochastic depth [19], RCNN [31], GoogleNet [43], etc. It is crucial to note that these methods are still indeed feedforward since iterative injection of thus-far output into the system would be essential for forming a proper feedback. This is a primary difference between our approach and many existing works. We empirically show that the feedback mechanism, besides the recurrence, is indeed critical for achieving the discussed advantages (Table 4).

A few recent encouraging methods explicitly employed feedback connections [6, 3, 45, 30, 32, 21] with promising results for the task of interest. The majority of these methods are either task specific and/or model temporal problems. They also do not put forth and investigate the core advantages of a general feedback based inference. We should also emphasize that feedback in our model is always within the hidden space. This will allow us to develop generic feedback based architectures without the requirement of task-specific error-to-input functions [6].

Another group of methods, mostly in sequence modeling literature, developed feedback like mechanism in order to control the spatial selectivity of a model [46, 5, 35]. This is usually used for better modeling of long term dependencies, computational efficiency, or spatial localization. This category of methods are different from ours as altering spatial selectivity does not have a clear contribution to many static tasks and the discussed general advantages of feedback.

Lastly, it is worth noting that Curriculum Learning [11, 26, 4] and making predictions on a taxonomy [18, 41, 8, 10, 22] are well investigated in the literature, though none provided a feedback based approach which is our focus.

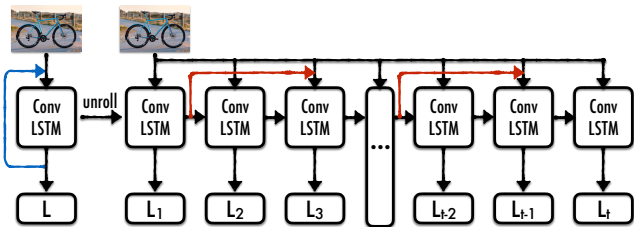

Figure 2. **Illustration of our core feedback model and skip connections (shown in red) when unrolled in time.** 'ConvLSTM' and 'L' boxes represent convolutional operations and iteration losses, respectively.

3. Feedback Networks

Feedback based prediction has two requirements: (1) iterativeness and (2) rerouting a notion of posterior (output) into the system in each iteration. We realize this by adopting a convolutional recurrent neural network model and connecting the loss to each iteration. The overall process can be summarized as: the image undergoes a shared convolutional operation repeatedly and a prediction is made at each time; the recurrent convolutional operations are trained to produce the best output at each iteration given a hidden state that carries a direction notation of thus-far output. This is depicted in Fig. 2.

3.1. Convolutional LSTM Formulation

In this section, we share the details of our feedback model which is based on stacking ConvLSTM [45] modules that essentially replace the operations in an LSTM [16] cell with convolutional structures [1]. An LSTM cell uses hidden states to pass information through iterations. We briefly describe the connections between stacked ConvLSTMs and the gates in them:

We parametrize the temporal order (i.e. iterations) with time $t = 0, 1, ..., T$ and spatial order of a ConvLSTM module in the stack with depth $d = 0, 1, ..., D$. At depth d and time t, the output of a ConvLSTM module is based on spatial input ($\mathbf{X}_t^{d-1}$), temporal hidden state input ($\mathbf{H}_{t-1}^d$), and temporal cell gate input ($\mathbf{C}_{t-1}^d$).

To compute the output of a ConvLSTM module, the input gate i_t^d and forget gate f_t^d are used to control the information passing between hidden states:

$$\begin{aligned} i_t^d &= \sigma(W_{d,xi}(\mathbf{X}_t^{d-1}) + W_{d,hi}(\mathbf{H}_{t-1}^d)), \\ f_t^d &= \sigma(W_{d,xf}(\mathbf{X}_t^{d-1}) + W_{d,hf}(\mathbf{H}_{t-1}^d)), \end{aligned} \tag{1}$$

where σ is sigmoid function. W is a set of feedforward convolutional operations applied to $\mathbf{X}$ and $\mathbf{H}$. Here W is parametrized by d but not t since the weights of convolutional filters are shared in the temporal dimension. The architecture of W is a design choice, and its depth (i.e. the physical depth of a ConvLSTM module) is discussed in Sec. 3.2.

The cell gate $\mathbf{C}_t^d$ is computed as follows:

$$\begin{aligned} \tilde{C}_t^d &= tanh(W_{d,xc}(\mathbf{X}_t^{d-1}) + W_{d,hc}(\mathbf{H}_{t-1}^d)), \\ \mathbf{C}_t^d &= f_t^d \circ \mathbf{C}_{t-1}^d + i_t^d \circ \tilde{C}_t^d. \end{aligned} \tag{2}$$

Finally, the hidden state $\mathbf{H}_t^d$ and output $\mathbf{X}_t^d$ are updated according to the output state o_t and cell state $\mathbf{C}_t^d$:

$$\begin{aligned} o_t^d &= \sigma(W_{d,xo}(\mathbf{X}_t^{d-1}) + W_{d,ho}(\mathbf{H}_{t-1}^d)), \\ \mathbf{H}_t^d &= o_t^d \circ tanh(\mathbf{C}_t^d), \\ \mathbf{X}_t^d &= \mathbf{H}_t^d, \end{aligned} \tag{3}$$

where '$\circ$' denotes the Hadamard product. Also, we apply batch normalization [20] to each convolutional operation.

For every iteration, loss is connected to the output of the last ConvLSTM module in physical depth. Here, the post processes of ConvLSTM module's output (pooling, fully connected layer, etc.) are ignored for sake of simplicity. $\mathbf{L_t}$ is the cross entropy loss at time t, while C denotes the correct target class number and $\mathbf{L}$ is the overall loss:

$$\mathbf{L} = \sum_{t=1}^{T} \gamma^t \mathbf{L_t}, \text{ where } \mathbf{L_t} = -log \frac{e^{\mathbf{H_t^D}[C]}}{\sum_j e^{\mathbf{H_t^D}[j]}}. \tag{4}$$

γ is a constant discount factor determining the worth of early vs later predictions; we set $\gamma = 1$ in our experiments which gives equal worth to all iterations. [2]

Connecting the loss to all iterations forces the network to attempt the entire task at each iteration and pass the output via the proxy of hidden state to future iterations (Eq. 4). Therefore, the network cannot adopt a representation scheme like feedforward networks that go from low-level (e.g. edges) to high-level representations since merely low-level representations would not be sufficient for accomplishing the whole classification task in the early iterations. This yields a network that forms a representation across iterations in a coarse-to-fine manner (further discussed in sections 4.2.2 and 4.2.3 as well as supplementary material's Sec. 2).

We initialize all $\mathbf{X_0^0}$ as the inout image inp, and all $\mathbf{H_0^d}$ as 0, i.e. $\forall t \in \{1, 2, \cdots, T\} : \mathbf{X}_t^0 := inp$ and $\forall d \in \{1, 2, \cdots, D\} : \mathbf{H}_0^d := 0$. The operation of the ConvLSTM module above can be referred to using the simplified notation $\mathfrak{F}(\mathbf{X}_t^{d-1}, \mathbf{H}_{t-1}^d)$.

[1] See supplementary material (Sec. 5) for a discussion on alternatives to LSTM for this purpose, including GRU, RNN, and ablated LSTM.

[2] **Predicting the 'absolute output' vs an 'adjustment' value**: In this formulation, we predict the absolute value of the output at each iteration. An alternative would be to predict an 'adjustment value' at each iteration that, when summed with previous iteration's output, would yield the updated absolute output. This approach would have the disadvantage of being applicable to only output spaces with a numerical structure, e.g. regression problems. Problems without such a structure, e.g. classification, cannot not be solved using this approach.

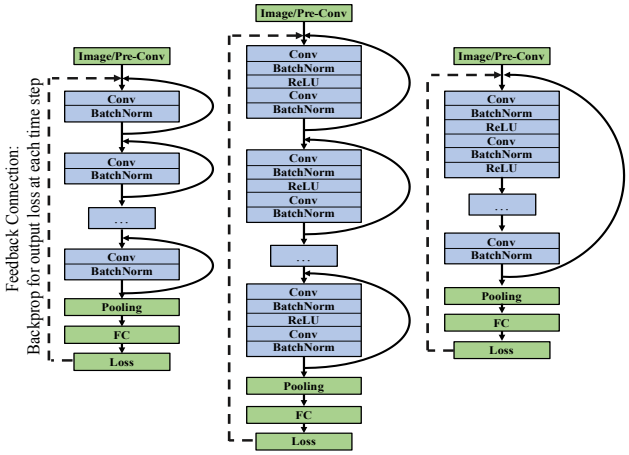

Figure 3. **Feedback networks with different feedback module (ConvLSTM) lengths. Left, middle, and right show Stack-1, Stack-2, and Stack-All, respectively.**

3.2. Feedback Module Length

The described architecture is flexible on the number of feedforward layers within a ConvLSTM module and the number of such ConvLSTM modules to stack. We can categorize the feedback networks according to the number of feedforward layers (Conv + BN) within one ConvLSTM module, i.e. the local length of feedback. This is shown in Fig. 3 where (a), (b) and (c) are named as Stack-1, Stack-2, and Stack-All. For a Stack-i module, i feedforward layers are stacked within the ConvLSTM. This essentially determines how distributed the propagation of hidden state throughout the network should be (e.g. for the physical depth $\mathcal{D}$, Stack-All architecture would have one hidden state while Stack-1 would have $\mathcal{D}$ hidden states). Which length i to pick is a design choice; we provide an empirical study on this in Sec. 4.2.1.

3.3. Temporal Skip Connection

In order to regulate the flow of signal through the network, we include identity skip connections. This was inspired by conceptually similar mechanisms, such as the residual connection of ResNet [14] and the recurrent skip coefficients in [48]. The skip connections adopted in the feedback model can be formulated as: with the new input at time t being $\hat{\mathbf{X}}_t^d = \mathbf{X}_t^d + \mathbf{H}_{t-n}^d$, the final representation will be $\mathfrak{F}(\hat{\mathbf{X}}_t^d, \mathbf{H}_{t-n}^d, \mathbf{H}_{t-1}^d)$, where n is the skip length. The skip connections are shown in Fig.2 denoted by the red dashed lines. We set $n = 2$ in our experiments.

Besides regulating the flow, Table 1 quantifies the endpoint performance improvement made by such skip connections on CIFAR100 [24] using Stack-2 architecture with physical depth 4 and 8 iterations.

3.4. Taxonomic Prediction

It is of particular practical value if the predictions of a model conform to a taxonomy. That is, making a correct coarse prediction about a query, if a correct fine prediction cannot be made. Given a taxonomy on the labels (e.g., ImageNet or CIFAR100 taxonomies), we can examine a network's capacity in making taxonomic predictions based on the fine class's Softmax distribution. The probability of a query belonging to the fine class y_i is defined in Softmax as $P(y_i|x;W) = \frac{e^{f_{y_i}}}{\sum_j e^{f_j}}$ for a network with weights W. The probability of a query belonging to the k^{th} higher level coarse class Y_k consisting of $\{y_1, y_2, ..., y_n\}$ is thus the sum of probability of the query being in each of the fine classes:

$$P(Y_k|x;W) = \sum_{i \in 1:n} P(y_i|x;W) = \frac{\sum_{i \in 1:n} e^{f_{y_i}}}{\sum_j e^{f_j}}. \quad (5)$$

Therefore, we use a mapping matrix M, where $M(i,k) = 1$ if $y_i \in Y_k$, to transform fine class distribution to coarse class distribution. This also gives us the loss for coarse prediction L^{Coarse}, and thus, a coarse prediction p_c is obtained through the fine prediction p_f. In Sec. 4.2.3, it will be shown that the outputs of the feedback network conform to a taxonomy especially in early predictions.

Feedback Net	Top1	Top5
w/o skip connections	67.37	89.97
w/ skip connections	**67.83**	**90.12**

Table 1. **Impact of skip connections in time on CIFAR100** [24]

3.5. Episodic Curriculum Learning

As discussed in Sec. 1, the feedback network provides a new way for enforcing a curriculum in learning and enables using a taxonomy as a curriculum strategy. We adopt an iteration-varying loss to enforce the curriculum. We use an annealed loss function at each time step of our k-iteration feedback network, where the relationship of coarse class losses L_t^{Coarse} and fine class losses L_t^{Fine} parametrized by time t is formulated as:

$$L(t) = \zeta L_t^{Coarset} + (1-\zeta) L_t^{Fine}, \quad (6)$$

where ζ is the weights that balance the contribution of coarse and fine losses. We adopt a linear decay as $\zeta = \frac{t}{k}$, where $t = 0, 1, 2, ..., k$, and k is the iteration at which we stop annealing.

For object classification, the time varying loss function encourages the network to recognize objects in a first coarse then fine manner, i.e. the network learns from the root of an taxonomy tree to its leaves. In Sec. 4.2.4, it will be empirically shown that the feedback based approach well utilizes this curriculum strategy.

3.6. Computation Graph Analysis

Under proper hardware, the feedback model also has an advantage on speed over feedforward. This is because a

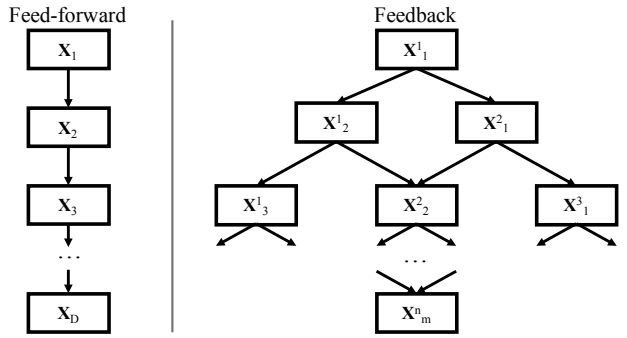

Figure 4. **Computation graph of Feedback vs Feedforward.** Skip connections are not shown for simplicity.

feedback network is a better fit for parallelism compared to a feedforward one due to having a shallower computation graph.[3] In the following paragraphs, we will discuss the computation graph depth of feedforward model with depth $\mathcal{D}$ and that of feedback model with same virtual depth (consisting of m temporal iterations and physical depth n, $\mathcal{D} = m \times n$). Feedforward computation has the limitation that representation $\mathbf{X}$ at depth i is dependent on the previous representation at depth $i-1$ creating a nested serial process, i.e. $\mathbf{X}^i$ depends on $\mathbf{X}^{i-1}$).

In our recurrent model, representation $\mathbf{X}_i^j$ at temporal iteration i and physical depth j is dependent on two representations: the previous iteration $\mathbf{X}_{i-1}^j$ (for stacked ConvLSTM, $\mathbf{H}_t^d = \mathbf{X}_\mathbf{t}^\mathbf{d}$, per Eq. 3) and the previous depth $\mathbf{X}_i^{j-1}$:

$$\mathbf{X}_i^j = \mathfrak{F}(\mathbf{X}_i^{j-1}, \mathbf{X}_{i-1}^j).$$

The resulting computation graphs of feedforward and feedback are shown in Fig. 4; notice that although both graphs have the same node count, they have different depths (longest directed path in a graph): feedforward's depth is $d_{ff} = \mathcal{D} - 1 = mn - 1$ while feedback network has depth $d_{fb} = m + n - 1$. In a proper hardware scenario where one can do parallel computations to a sufficient extent, inference time can be well measured by the longest distance from root to target (same as the graph's depth). An example of feedback network's computation can be seen in Fig. 5. Notice that for our computation, we have $max(m, n)$ parallel computations at each time step. Therefore, the total prediction time of feedforward network is larger than feedback network's since $d_{ff} = mn - 1 > m + n - 1 = d_{fb}$.

The run-time for early iteration's prediction can also be measured by the longest distance from root to the output at k^{th} iteration. For feedback network, the distance $d_{ff_k} = n + k - 1$, but for feedforward network $d_{fb_k} = nk - 1$. We have $d_{ff_k} = nk - 1 > k + n - 1 = d_{fb_k}$.

It is worth noting that the practical run-time of an algorithm depends on various factors, such as implementation and hardware. The purpose of the above analysis on computation graph's depth is to exclude the impact of those factors and quantify the true capacity of each method. The above derivation is applicable to training time as well, given proper hardware.

[3]The number of layers, number of parameters, etc are not proper metrics for an algorithm's run-time.

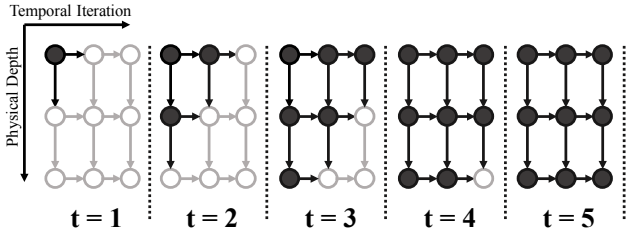

Figure 5. **Illustration of computation of feedback's inference pass.** Each node represents a ConvLSTM module. Skip connections are not shown for simplicity.

4. Experimental Results

Our experimental evaluations performed on the three benchmarks of CIFAR100 [24], Stanford Cars [23], and MPII Human Pose [1], are provided in this section.

4.1. Baselines and Terminology

Below we define our terminology and baselines:

Physical Depth: the depth of the convolutional layers from input layer to output layer. For feedback networks, this represents the number of stacked physical layers across all ConvLSTM modules ignoring the temporal dimension.

Virtual Depth: physical depth $\times$ number of iterations. This is the effective depth considering both spatial and temporal dimensions. (not applicable to feedforward models.)

Baseline Models: We compare with ResNet[14] and VGG[39] as two of the most commonly used feedforward models and with closest architecture to our convolutional layers. Both baselines have the same architecture, except for the residual connection. We use the same physical module architecture for our methods and the baselines. We also compare against ResNet original authors' architecture [14]. The kernel sizes and the transitions of filter numbers remain the same as original authors'. We also compare with feedforward Hourglass [36] architecture by making a feedback Hourglass in Sec. 4.4.

Auxiliary prediction layer (aux loss): Feedfoward baselines do not make episodic or mid-network predictions. In order to have a feedforward based baseline for such predictions, we train new pooling→FC→loss layers for different depths of the feedforward baselines (one dedicated aux layers for each desired depth). This allows us to make predictions using the mid-network representations. We train these aux layers by taking the fully trained feedforward network and training the aux layers from shallowest to deepest layer while freezing the convolutional weights.

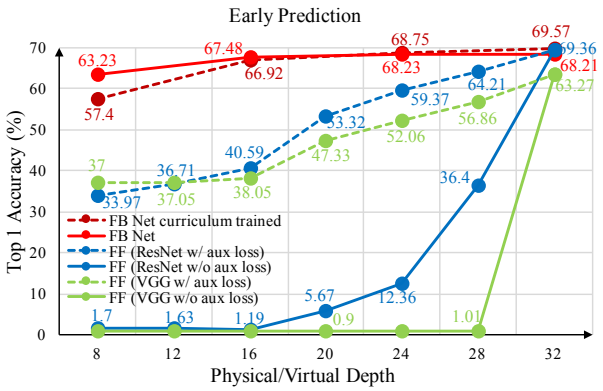

Figure 6. **Evaluation of early predictions.** Comparison of accuracy of feedback (FB) model and feedforward (FF) baselines (ResNet & VGG, with or without auxiliary loss layers)

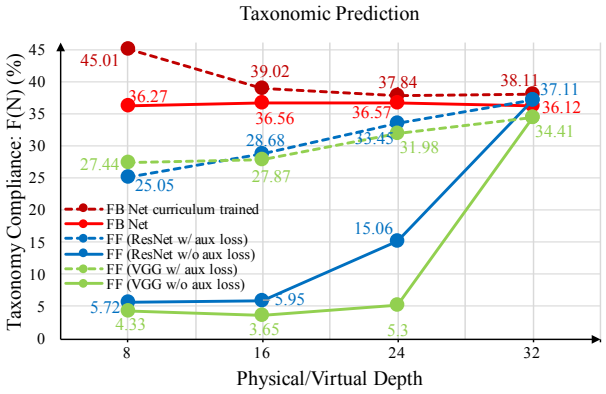

Figure 7. **Evalaution of taxonomoy based prediction for feedback (FB) and feedforward (FF) networks trained with or without auxiliary layers.** We use only fine loss to train, except for the curriculum learned one.

4.2. CIFAR-100 and Analysis

CIFAR100 includes 100 classes containing 600 images each. The 100 classes are categorized into 20 classes. The original 100 classes are the fine level labels and the 20 superclasses are the coarse ones, forming a 2-level taxonomy.

4.2.1 Feedback Module Length

We first study the effect of feedback module length per the discussion in Sec. 3.2. Table 2 provides the results of this test while the physical depth and iteration count are kept constant (physical depth 4 and 4 iterations) for all models.

The best performance is achieved when the local feedback length is neither too short nor too long. We found this observation to be valid across different tests and architectures, though the optimal length may not always be 2. In the rest of the experiments for different physical depths, we optimize the value of this hyperparameter empirically (often ends up as 2 or 3). See supplementary material's Sec. 4 for discussions on performance vs physical depth as well as optimal iteration number.

Feedback Type	Top1	Top5
Stack-1	66.29	89.58
Stack-2	**67.83**	**90.12**
Stack-All	65.85	89.04

Table 2. **Comparison of different feedback module lengths,** all models have the same physical depth 4 and virtual depth 16

4.2.2 Early Prediction

We evaluate early predictions of various networks in this section. We conduct this study using a feedback network with virtual depth 32 (similar trends achieved with other depths) and compare it with various feedforward networks. As shown in Fig. 6, at virtual depths of 8, 12, and 16, the feedback network already achieves satisfactory and increasing accuracies. The solid blue and green curves denote the basic feedforward networks with 32 layers; their rightmost performance is their endpoint results, while their early predictions are made using their final pooling→FC→loss layer but applied on mid-network representations. The dashed blue and green curves show the same, with the difference that the trained pooling→FC→loss layers (aux loss, described in Sec. 4.1) are employed for making early predictions. The plot shows that the feedforward networks perform poorly when using their first few layers' representations, confirming that the features learned there are not suitable for completing the ultimate output task (expected). This is aligned with the hypothesis that a deep feedforward network is forming its representation from low-level features (e.g. edges) to high-level abstractions [47], while the feedback model forms its representation in a different and coarse-to-fine manner (further discussed in Sec. 4.2.3).

	Time Steps			
Model	12T	15T	18T	21T
Feedback Network	**67.94**	**70.57**	**71.09**	**71.12**
ResNet Ensemble	66.35	67.52	67.87	68.2

Table 3. **Top1 accuracy comparison between Feedback Net and an ensemble of ResNets** that produce early predictions at the same computation graph depth time steps.

Comparison with Feedforward Ensemble: Although it is memory inefficient and wasteful in training, one can also achieve the effect of early prediction through an ensemble of feedforward models in parallel (i.e. for every depth at which one desires a prediction, have a dedicated feedforward network of that depth). Since running an ensemble of ResNets in parallel has similar optimal hardware requirements of Sec. 3.6, we make a comparison under the same analysis: applying the computation graph depth analysis to a 48 layer virtual depth feedback model (physical depth 12, Stack-3, 4 iterations), if we denote the time to finish one layer of convolution as T, then we have i^{th} iteration result at: $t_i = (12 + 3i)T$. Then the first to last iterations' results (virtual depths $12, 24, 36, 48$) will become available at $12T, 15T, 18T$, and $21T$. To have ResNet results at the same times, we need an ensemble of ResNets

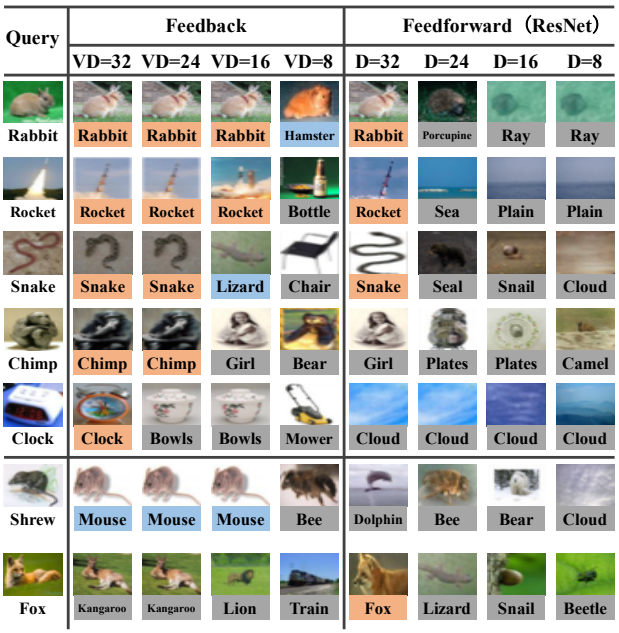

Figure 8. **Qualitative results of classification on CIFAR100.** Each row shows a query along with nearest neighbors at different depth for feedback and feedfowrad networks. Orange, blue, and gray represent 'correct fine class', 'correct coarse class but wrong fine class', and 'both incorrect', respectively. Two bottom queries are representative failure cases.

with depths $12, 15, 18, 21$. The performance comparison between feedback network and the ensemble is provided in Table 3, showing the advantage of feedback networks.

Feedback vs No Feedback: To examine whether the offered observations and advantages are caused by feedback or only the recurrence mechanism, we performed a test by disconnecting the loss from all iterations except the last, thus making the model recurrent-feedforward. As shown in Table 4, making the model recurrent-feedforward takes away the ability to make early predictions and taxonomic predictions (discussed next).

	Virtual Depth			
Model	12	24	36	48
Feedback	**67.94**	**70.57**	**71.09**	71.12
Feedback Disconnected	36.23	62.14	67.99	**71.34**

Table 4. **The impact of feedback** on CIFAR100 for a model with virtual depth 48 and four iterations.

4.2.3 Taxonomic Prediction

We measure the capacity $F(N)$ of network N in making taxonomic predictions (taxonomy compliance) as: the probability of making a correct coarse prediction for a query if it made a wrong fine prediction for it; in other words, how effective it can correct its wrong fine class prediction to a correct coarse class: $F(N) = P(correct(p_c)|!correct(p_f); N)$. As defined in Sec. 3.4, pc and pf stand for coarse and fine prediction, respectively.

The quantitative and qualitative results are provided in Figures 7 and 8, respectively. Note that the results shown in both figures were naturally achieved, i.e. no taxonomy was used during training (except for the dashed red curve which was trained using curriculum learning; Sec. 4.2.4). Fig. 7 shows feedback network's predictions better complies with a taxonomy even at shallow virtual depths, while feedforward mdoel does not achieve the same performance till the last layer, even when using dedicated auxiliary layers. This is again aligned with the hypothesis that the feedback based approach develops a coarse-to-fine representation and is observed in both figures 8 and 9. In Fig. 8, early prediction classes and nearest neighbor images (using the network representations) for both feedback and feedforward networks are provided, showing significantly more relevant and interpretable early results for feedback.

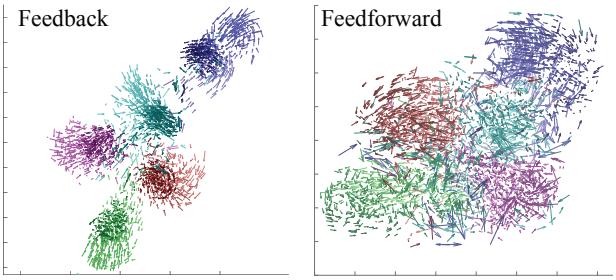

Figure 9. **Timed-tSNE** plots for five random CIFAR100 classes showing how the representation evolves through depth/iterations for each method (i.e. how a datapoint moved in representation space). The brighter the arrow, the earlier the depth/iteration. Feedback's representation is relatively disentangled throughout, while feedforward's representation gets disentangled only towards the end. (Best see on screen. Vector lengths are shown in half to avoid cluttering.)

Timed-tSNE: In Fig. 9, we provide a variant of tSNE [34] plot which we call *timed-tSNE*. It illustrates *how the representation of a network evolves* throughout depth/iterations, when viewed through the window of class labels. For each datapoint, we form a regulated trajectory by connecting a set of 2D tSNE embedding locations. For feedback network, the embeddings of one datapoint come from the representation at different iterations (i.e. i embeddings for a network with i iterations). For feedforward, embeddings come from difference layers. More details provided in supplementary material (Sec. 3).

Fig. 9 suggests that feedforward representation is intertwined at early layers and disentangles the classes only in the last few layers, while feedback's representation is disentangled early on and the updates are mostly around forming fine separation regions. This again supports the hypothesis that feedback develops a coarse-to-fine representation. We also provide activation maps of feedback vs feedforward models in supplementary material (Sec. 3.2) showing notably dissimilar patterns, and therefore, dissimilar representations.

4.2.4 Curriculum Learning

Table 5 compares the performance of the networks when trained with the fine-only loss vs the episodic coarse-to-fine curriculum loss (Sec. 3.5). We employed the same episodic curriculum training for the feedback network and the baselines "w/ Aux loss", while the baselines "w/o Aux loss" had to use conventional curriculum training (datapoint sorting) [4]. The best performance with highest boost is achieved by feedback network when using curriculum learning. This suggests that the feedback based approach best fits episodic curriculum training, as discussed in Sec. 1. Also, using the episodic curriculum training improves taxonomic prediction results as shown by the curriculum curve in Fig. 7.

Model	CL	Top1(%)-Fine	Top1(%)-Coarse
Feedback Net	N	68.21	79.7
	Y	**69.57**(+1.34%)	**80.81**(+1.11%)
Feedforward ResNet w/ Aux loss	N	69.36	80.29
	Y	69.24(-0.12%)	80.20(-0.09%)
Feedforward ResNet w/o Aux loss	N	69.36	80.29
	Y	65.69(-3.67%)	76.94(-3.35%)
Feedforward VGG w/ Aux loss	N	63.56	75.32
	Y	64.62(+1.06%)	77.18(+1.86%)
Feedforward VGG w/o Aux loss	N	63.56	75.32
	Y	63.2(-0.36%)	74.97(-0.35%)

Table 5. **Evaluation of the impact of Curriculum Learning (CL)** on CIFAR100. The CL column denotes if curriculum learning was used. The difference made by curriculum for each method is shown in parentheses.

4.2.5 Endpoint Performance Comparison

Table 6 compares the endpoint performance of various feedforward and feedback models on CIFAR100. The detailed architecture of each model is provided in the end of this section. Feedback networks outperform the baselines with the same physical depth by a large margin and work better than or on part with baselines with the same virtual depth or deeper. This ensures that the discussed advantages in early and taxonomic prediction were not achieved at the expense of sacrificing the endpoint performance.

The bottom part of Table 6 shows several recent methods that are not comparable to ours, as they employ additional mechanisms (e.g. stochasticity in depth [19]) which we did not implement in our model. Such mechanisms are independent of feedback and could be used concurrently with it, in the future. However, we include them for the sake of completeness.

Architectures: The detailed architectures of feedback and feedforward networks are: [4]

[4]The following naming convention is used: $C(fi, fo, k, s)$: fi input and fo output convolutional filters, kernel size $k \times k$, stride s. $ReLU$: rectified linear unit. BN: batch normalization. $BR = BN + ReLU$. $Avg(k, s)$: average pooling with spatial size $k \times k$, and stride s. $FC(fi, fo)$: fully connected layer with fi inputs, and fo outputs.

Model	P/D	V/D	Top1 (%)	Top5 (%)
Feedforward (ResNet[14]) 	48	-	70.04	90.96
	32	-	69.36	91.07
	12	-	66.35	90.02
	8	-	64.23	88.95
	128*	-	70.92	91.28
	110*	-	72.06	92.12
	64*	-	71.01	91.48
	48*	-	70.56	91.60
	32*	-	69.58	91.55
Feedforward (VGG[39]) 	48	-	55.08	82.1
	32	-	63.56	88.41
	12	-	64.65	89.26
	8	-	63.91	88.90
Feedback Net 	12	48	**71.12**	**91.51**
	8	32	69.57	91.01
	4	16	67.83	90.12
Highway [42]	19	-	67.76	-
ResNet V2[15]	1001	-	77.29	-
Stochastic Depth [19]	110	-	75.02	-
SwapOut [40]	32 fat	-	77.28	-
RCNN [31]	4 fat	16	68.25	-

Table 6. **Endpoint performance comparison on CIFAR-100.** Baselines denoted with * are the architecture used in the original ResNet paper. P/D=physical depth, V/D=virtual depth.

• Recurrent Block: $Iterate(fi, fo, k, s, n, t)$ denotes the recurrent modules in the feedback networks which iterates t times through the feedforward blocks therein:
$\rightarrow C(fi, fo, k, s) \rightarrow BR \rightarrow \{C(fo, fo, k, 1) \rightarrow BR\}^{n-1}$
we denote stacking using $\{...\}^n$ indicating that the module in the bracket is stacked n times.
• Preprocess and Postprocess: across all models, we apply the following pre-process: $Input \rightarrow C(3, 16, 3, 1) \rightarrow BR$ and post-process: $\rightarrow Avg(8, 1) \rightarrow FC(64, 100)$
• Feedback Network with physical depth = 8:
$\rightarrow Iterate(16, 32, 3, 2, 2, 4) \rightarrow Iterate(32, 32, 3, 1, 2, 4)$
$\rightarrow Iterate(32, 64, 3, 2, 2, 4) \rightarrow Iterate(64, 64, 3, 1, 2, 4)$
• Feedback Network with physical depth = 12:
$\rightarrow Iterate(16, 16, 3, 1, 3, 4) \rightarrow Iterate(16, 32, 3, 2, 3, 4)$
$\rightarrow Iterate(32, 64, 3, 2, 3, 4) \rightarrow Iterate(64, 64, 3, 1, 3, 4)$
• Baseline Feedforward models with physical depth = $\mathcal{D}$:
$\rightarrow C(16, 32, 3, 2) \rightarrow BR \rightarrow \{C(32, 32, 3, 1) \rightarrow BR\}^{\frac{\mathcal{D}}{2}-1}$
$\rightarrow C(32, 64, 3, 2) \rightarrow BR \rightarrow \{C(64, 64, 3, 1) \rightarrow BR\}^{\frac{\mathcal{D}}{2}-1}$

4.3. Stanford Cars Dataset

To verify the observations made on CIFAR100 on another dataset, we performed the same set of experiments on Stanford Cars dataset [23]. Evaluations of endpoint performance and curriculum learning are provided in table 7. Early prediction and taxonomic prediction curves are provided in supplementary material (Sections 6.1 and 6.2). The

experiments show similar trends to CIFAR100's and duplicate the same observations.

Model	CL	Fine	Coarse
Feedback Net	N	50.33	74.15
	Y	**53.37**(+3.04%)	**80.7**(+6.55%)
Feedforward ResNet-24	N	49.09	72.60
	Y	50.86(+1.77%)	77.25(+4.65%)
Feedforward VGG-24	N	41.04	67.65
	Y	41.87(+0.83%)	70.23(+2.58%)

Table 7. **Evaluations on Stanford Cars dataset**. The CL column denotes if curriculum learning was employed. All methods have (virtual or physical) depth of 24.

All networks were trained from scratch without finetuning pretrained ImageNet [9] models [33] or augmenting the dataset with additional images [44]. To suit the relatively smaller amount of training data in this dataset, we use shallower models for both feedforward and feedback: feedforward baselines have depth of 24 and feedback network has physical depth 6 and iteration count 4, following the same design in Sec. 4.1 & Sec. 4.2.5. Full experimental setup details are provided in supplementary material (Sec. 6).

4.4. Human Pose Estimation

We evaluated on the regression task of MPII Human Pose estimation [1] benchmark which consists of 40k samples (28k training, 11k testing). Just like we added feedback to feedforward models for CIFAR100 classification and performed comparisons, we applied feedback to the state of the art MPII model Hourglass [36]. We replaced the sequence of ResNet-like convolutional layers in one stack Hourglass with ConvLSTM, which essentially repalced physical depth with virtual depth, and performed backprobapation at each iteration similar to the discussion in Sec. 3.1 (more details about the architecture provided in supplementary material). The performance comparison in Table 8 shows that the feedback model outperform the comparable and deeper feedforward baseline. We provide more results and comparisons with other feedback based methods [6, 3] on this benchmark in supplementary material (Sec. 7).

Method	P/D	V/D	PCKh
Feedforward-Hourglass	24	-	77.6
Feedback-Hourglass	4	12	**82.3**

Table 8. **Evaluations on MPII Human Pose Dataset**

5. Conclusion

We provided a study on feedback based learning, showing it is a worthwhile alternative to heavily employed feedforward model with several basic advantages: early prediction, taxonomy compliance, and Episodic Curriculum Learning. We also observed that the feedback based approach develops a coarse-to-fine representation that is meaningfully and considerably different from feedforward representations. The study suggests it would not be farfetched to find the best practices of computer vision lying in a feedback based approach, rather than feedforward, in the near future. We hope this study provides the community with new perspectives and motivation to investigate this paradigm.

References

[1] M. Andriluka, L. Pishchulin, P. Gehler, and B. Schiele. 2d human pose estimation: New benchmark and state of the art analysis. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3686–3693. IEEE, 2014.

[2] S. J. Ashford and L. L. Cummings. Feedback as an individual resource: Personal strategies of creating information. *Organizational behavior and human performance*, 32(3):370–398, 1983.

[3] V. Belagiannis and A. Zisserman. Recurrent human pose estimation. *arXiv preprint arXiv:1605.02914*, 2016.

[4] Y. Bengio, J. Louradour, R. Collobert, and J. Weston. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pages 41–48. ACM, 2009.

[5] C. Cao, X. Liu, Y. Yang, Y. Yu, J. Wang, Z. Wang, Y. Huang, L. Wang, C. Huang, W. Xu, et al. Look and think twice: Capturing top-down visual attention with feedback convolutional neural networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2956–2964, 2015.

[6] J. Carreira, P. Agrawal, K. Fragkiadaki, and J. Malik. Human pose estimation with iterative error feedback. *arXiv preprint arXiv:1507.06550*, 2015.

[7] R. M. Cichy, D. Pantazis, and A. Oliva. Resolving human object recognition in space and time. *Nature neuroscience*, 17(3):455–462, 2014.

[8] J. Deng, N. Ding, Y. Jia, A. Frome, K. Murphy, S. Bengio, Y. Li, H. Neven, and H. Adam. Large-scale object classification using label relation graphs. In *European Conference on Computer Vision*, pages 48–64. Springer, 2014.

[9] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 248–255. IEEE, 2009.

[10] N. Ding, J. Deng, K. P. Murphy, and H. Neven. Probabilistic label relation graphs with ising models. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1161–1169, 2015.

[11] J. L. Elman. Learning and development in neural networks: The importance of starting small. *Cognition*, 48(1):71–99, 1993.

[12] F. A. Ford. *Modeling the environment: an introduction to system dynamics models of environmental systems*. Island Press, 1999.

[13] D. Ha, A. Dai, and Q. V. Le. Hypernetworks. *arXiv preprint arXiv:1609.09106*, 2016.

[14] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385*, 2015.

[15] K. He, X. Zhang, S. Ren, and J. Sun. Identity mappings in deep residual networks. *arXiv preprint arXiv:1603.05027*, 2016.

[16] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, Nov. 1997.

[17] C. S. Holling. Resilience and stability of ecological systems. *Annual review of ecology and systematics*, pages 1–23, 1973.

[18] H. Hu, G.-T. Zhou, Z. Deng, Z. Liao, and G. Mori. Learning structured inference neural networks with label relations. *arXiv preprint arXiv:1511.05616*, 2015.

[19] G. Huang, Y. Sun, Z. Liu, D. Sedra, and K. Weinberger. Deep networks with stochastic depth. *arXiv preprint arXiv:1603.09382*, 2016.

[20] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[21] A. Karpathy and L. Fei-Fei. Deep visual-semantic alignments for generating image descriptions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3128–3137, 2015.

[22] D. Koller and N. Friedman. *Probabilistic graphical models: principles and techniques*. MIT press, 2009.

[23] J. Krause, M. Stark, J. Deng, and L. Fei-Fei. 3d object representations for fine-grained categorization. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pages 554–561, 2013.

[24] A. Krizhevsky and G. Hinton. Learning multiple layers of features from tiny images. 2009.

[25] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, 2012.

[26] K. A. Krueger and P. Dayan. Flexible shaping: How learning in small steps helps. *Cognition*, 110(3):380–394, 2009.

[27] S. Kumar and W. Byrne. Minimum bayes-risk decoding for statistical machine translation. Technical report, DTIC Document, 2004.

[28] E. B. Lee and L. Markus. Foundations of optimal control theory. Technical report, DTIC Document, 1967.

[29] T. S. Lee and D. Mumford. Hierarchical bayesian inference in the visual cortex. *JOSA A*, 20(7):1434–1448, 2003.

[30] K. Li, B. Hariharan, and J. Malik. Iterative instance segmentation. *arXiv preprint arXiv:1511.08498*, 2015.

[31] M. Liang and X. Hu. Recurrent convolutional neural network for object recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3367–3375, 2015.

[32] Q. Liao and T. Poggio. Bridging the gaps between residual learning, recurrent neural networks and visual cortex. *arXiv preprint arXiv:1604.03640*, 2016.

[33] T.-Y. Lin, A. RoyChowdhury, and S. Maji. Bilinear cnn models for fine-grained visual recognition. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1449–1457, 2015.

[34] L. v. d. Maaten and G. Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(Nov):2579–2605, 2008.

[35] V. Mnih, N. Heess, A. Graves, et al. Recurrent models of visual attention. In *Advances in Neural Information Processing Systems*, pages 2204–2212, 2014.

[36] A. Newell, K. Yang, and J. Deng. Stacked hourglass networks for human pose estimation. *arXiv preprint arXiv:1603.06937*, 2016.

[37] A. G. Parlos, K. T. Chong, and A. F. Atiya. Application of the recurrent multilayer perceptron in modeling complex process dynamics. *IEEE Transactions on Neural Networks*, 5(2):255–266, 1994.

[38] N. C. Rust and J. J. DiCarlo. Selectivity and tolerance (???invariance???) both increase as visual information propagates from cortical area v4 to it. *The Journal of Neuroscience*, 30(39):12978–12995, 2010.

[39] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[40] S. Singh, D. Hoiem, and D. Forsyth. Swapout: Learning an ensemble of deep architectures. *arXiv preprint arXiv:1605.06465*, 2016.

[41] Y. Song, M. Zhao, J. Yagnik, and X. Wu. Taxonomic classification for web-based videos. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 871–878. IEEE, 2010.

[42] R. K. Srivastava, K. Greff, and J. Schmidhuber. Highway networks. *arXiv preprint arXiv:1505.00387*, 2015.

[43] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–9, 2015.

[44] S. Xie, T. Yang, X. Wang, and Y. Lin. Hyper-class augmented and regularized deep learning for fine-grained image classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2645–2654, 2015.

[45] S. Xingjian, Z. Chen, H. Wang, D.-Y. Yeung, W.-k. Wong, and W.-c. Woo. Convolutional lstm network: A machine learning approach for precipitation nowcasting. In *Advances in Neural Information Processing Systems*, pages 802–810, 2015.

[46] K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhutdinov, R. S. Zemel, and Y. Bengio. Show, attend and tell: Neural image caption generation with visual attention. *arXiv preprint arXiv:1502.03044*, 2(3):5, 2015.

[47] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In *European Conference on Computer Vision*, pages 818–833. Springer, 2014.

[48] S. Zhang, Y. Wu, T. Che, Z. Lin, R. Memisevic, R. Salakhutdinov, and Y. Bengio. Architectural complexity measures of recurrent neural networks. *arXiv preprint arXiv:1602.08210*, 2016.