MID-VISION FEEDBACK FOR CONVOLUTIONAL NEURAL NETWORKS

Anonymous authors
Paper under double-blind review

ABSTRACT

Feedback plays a prominent role in biological vision, where perception is modulated based on agents' continuous interactions with the world, and evolving expectations and world model. We introduce a novel mechanism which modulates perception in Convolutional Neural Networks (CNNs) based on high level categorical expectations: Mid-Vision Feedback (MVF). MVF associates high level contexts with linear transformations. When a context is "expected" its associated linear transformation is applied over feature vectors in a mid level of a CNN. The result is that mid-level network representations are biased towards conformance with high level expectations, improving overall accuracy and contextual consistency. Additionally, during training mid-level feature vectors are biased through introduction of a loss term which increases the distance between feature vectors associated with different contexts. MVF is agnostic as to the source of contextual expectations, and can serve as a mechanism for top down integration of symbolic systems with deep vision architectures - applications range from image and video understanding to explainable AI and robotics. We show the superior performance of MVF to post-hoc filtering for incorporation of contextual knowledge, and show superior performance of configurations using predicted context (when no context is known a priori) over configurations with no context awareness.

1 INTRODUCTION

In most contemporary computer vision architectures information flows in a single direction: from low-level of pixels up to a high level abstract concepts (e.g., object categories) - such architectures are termed *feed-forward* architectures. In general, each successive layer of the network contains more abstract representations than the previous, and the representational hierarchy mirrors the architectural hierarchy. It is also possible to introduce *top-down connections* in the network architecture, introducing high level information into processes involving lower levels of abstraction in a process of *feedback*.

Feedback plays a primary role in biological vision; in fact, the majority of neural connections in the human visual cortex are top-down, rather than bottom-up, connections (Kveraga et al., 2007). These top-down connections are thought to convey information of higher level *expectation*, and neurons of the visual cortex use both higher level expectation as well as lower level visual information in producing their representations. Expectations in biological systems arise from continuous engagement with the environment. In Computer Vision, this is reflected in the paradigm of Active Vision (Bajcsy, 1988), where perception is framed as an active problem involving evolving world models.

The task of producing mid-level visual representations from low level input is under-constrained - many plausible mid-level interpretations may be consistent with input. To give an intuition for how understanding of context can impact perception of mid-level features consider Figure 1 - characteristics of shrews and kiwi differ, but may be similar enough to be confused without context. Top-down feedback - from high level context to mid-level visual features - provides a "map" for mid-level processes, constraining it towards high level consistency.

Introduction of contextual knowledge through feedback is superior to post-hoc application of contextual knowledge, e.g. through discarding interpretations (classifications, here) which are not context consistent. We demonstrate this point empirically.

Figure 1: Here we illustrate images cropped to exclude context. At first glance, due to similarities in color, texture, and pattern, images from the top row (animals) may appear to be of the same class as those of the bottom row (food items). With an understanding of the difference of context, upon closer inspection, it is clear that there are meaningful lower level feature differences.

Interpretations selected after post-hoc filtering for context consistency will still be built upon under-constrained mid-level features. Furthermore, in contrast to post-hoc filtering, feedback naturally allows for detection of out-of-context objects (which, on occasion, occurs - e.g., a tree in an office setting), as feedback functions through biasing of visual representations rather than filtering.

CNNs have a natural tendency towards *decoupled* representations - representations with a tendency for feature vector angle correspond to characteristic type (e.g., "fuzzy"), and for feature vector magnitude corresponds to characteristic variation or degree (Liu et al., 2018) ("very fuzzy" / "not fuzzy") (See Figure 7 in the Appendix for an illustration). This opens up a couple possibilities in terms of directly manipulating feature representations: 1) We can differentiate between axes with different associations to high level contexts, 2) we can control magnitudes of characteristics through amplifying and dampening axes associated with those characteristics. That is, w.r.t. point #1, as CNNs produce representations which are, to a degree, separated by angle, certain axes will be more associated with some higher level contexts over others. w.r.t. point #2, amplifying characteristics associated with a higher level context increases the likelihood of interpreting input as conforming to that context; dampening characteristics associated with that context reduces the likelihood of interpreting input as conforming to that context.

We present a principled method to feedback - Mid-Vision Feedback (MVF), illustrated in Figure 2 - allowing the biasing of mid-level feature representations in networks such as CNNs towards conformance with high level categorical expectations. This approach is comprised of two components: 1) linear transforms (*affine transformations*), and 2) *orthogonalization bias*.

Affine transformations enable direct control over the feature vectors at the *injection level* - the level into which feedback is being inserted. The fact that these vectors have been trained with a bias towards orthogonality w.r.t. contexts means that, through affine transformations, characteristics associated with context presence or absence can be manipulated with less collateral impact on other features.

Orthogonalization bias introduces a tendency towards orthogonality at the level of the architecture into which feedback is to be introduced (the *injection level*). This does not negatively impact the representational power or performance of the base network. This orthogonality bias is introduced across contexts - e.g., mid-level feature vectors associated with *animate* contexts can be biased towards orthogonality w.r.t. mid-level features associated with *inanimate* contexts. Due to the resulting greater angular separation between features associated with different contexts, this biasing allows greater control over facets of mid-level representation which are meaningful to higher level contexts.

MVF then functions as follows. During runtime a high level contextual expectation is associated with input. This expectation is used in biasing mid-level visual features through use of an affine transformation associated with the context of that expectation. This selects for characteristics associated with this context. These affines are better enabled as a consequence of the disentanglement of such characteristics at the injection level, effected through introduction of an orthogonalization bias during training.

Feedback then enables a synergy between high level categorical interpretations and mid-level visual feature representations, bridging the signal-symbol gap in both directions. This approach to incorporation of context expectations is controlled. This differs from an approach of connecting the upper level fully connected layers of a network directly to lower level convolutional levels in a scheme which includes neither categorical representations nor biasing w.r.t. said categorical representations.

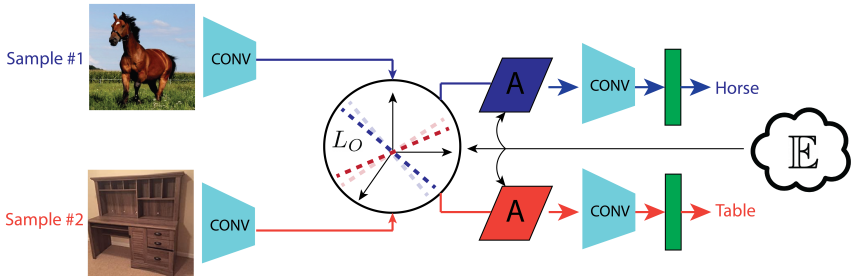


Figure 2: MVF structure. As training involves a contrastive loss between samples belonging to different contexts, the base network is shown twice in the figure: on the top when fed with a horse image, and on the bottom when fed with a desk image. *Affine transformations* (linear transformations) are inserted into a conventional CNN architecture. These affines modulate feature representations at *injection levels* (network levels over which affines are applied) in accordance with high level *context expectations* (E). A contrastive loss (L_O) is inserted to put pressure on the network to help separate feature representations at the injection levels according to context, better enabling affine manipulation. During training, samples from different contexts are fed through the network. The context expectations determine the behavior of L_0 and the selection of appropriate affines.

MVF both employs feedback from categorical knowledge and is agnostic w.r.t. the source of that categorical knowledge - i.e., it is not a requirement that context expectations be produced from the same network. As a consequence of this, our approach enables interfacing feedback with larger symbolic systems - e.g., models of scenes employing graphical models over scene elements and categories. This top-down synergy across the signal-symbol gap opens up a wide range of applications. This could include work on explainable AI, which may seek tighter integration of symbolic representations for the purpose of human interpretability. This also includes systems with dynamically changing input (e.g., video understanding or robotics domains) where the vision component can be dynamically updated to match an evolving understanding of the environment being perceived, or an evolving set of goals and associated relevant visual characteristics.

The rest of this paper is structured as follows: In Section 2 we cover related work; in Section 3 we detail methods; in Section 4 we cover experiments; and, in Section 5 we conclude.

2 Related Work

2.1 Feedback in Biology and Parellels to Computer Vision

Previous works ((Kveraga et al., 2007), (Gilbert & Sigman, 2007), Kreiman & Serre (2020), (Gilbert & Li, 2013), and (Paneri & Gregoriou, 2017)) have explored the importance of feedback connections in biological sensory perception. Further work ((Liao & Poggio, 2016) and (van Bergen & Kriegeskorte, 2020)) draw connections between feedback in computer vision architectures and the primate visual cortex.

There is good reason to believe that modeling characteristics of biological vision in computer vision architectures will benefit computer vision (e.g., (Medathati et al., 2016)). For example, (Linsley et al., 2020b) demonstrate a network with top-down connections which aligns with human perception of visual illusions, where feedback aids in prioritizing object boundary contours over simple edge contours. (Linsley et al., 2020a) shows how recurrent hierarchical feedback model can improve segmentation.

2.2 FEEDBACK IN EXISTING COMPUTER VISION METHODS

The conventional approach to feedback in computer vision is the use of recurrent connections. (Caswell et al., 2016) introduce recurrent connections into shallow CNN architecture for image classification. (Pinheiro & Collobert, 2014) employ recurrency over convolutions for the purpose of enabling lateral information flow in the task of segmentation. (Zamir et al., 2016) instantiate feedback through an RNN architecture which iteratively refines prediction categories from course to specific.

Alternatives to conventional recurrent connections for feedback include (Hu & Ramanan, 2016), which explore convolutions with hierarchical rectified Gaussians to enable top-down as well as bottom-up information flow, and apply them to the task of keypoint localization under occlusion. Additionallym, (Yao et al., 2012) apply a graphical model over scene representations, allowing higher and lower level decisions to influence each other.

Feedback has also been explored in methods which backtrace gradients in neural networks. One use case for this is localization of class relevant features - (Cao et al., 2015) and (Cao et al., 2018) implement an approach to feedback where activations are back-traced through the network, and demonstrate that this method is adept at localizing features of the classes for which feedback is applied. Approaches such as grad-CAM (Selvaraju et al., 2017) produce heatmaps over input images of regions relevant to a classification through backtracing gradients through the network.

3 METHODS

With MVF we seek a feedback mechanism which allows us to directly bias lower level feature representations based on categorical higher level context expectations. This involves top-down interaction across two levels of abstraction: 1) high level contexts $c_i \in C$, 2) mid-level features $f_i \in F$.

The structure of MVF is illustrated in Figure 2, and the loss and training formulation given in Section 3.1. Context expectations sit at a level of abstraction above the classes of network output, and are used in selecting affines placed above the output of injection levels. When applied to injection level output, the affines bias injection level feature representations towards conformance with the associated context expectation, as illustrated in Figure 3. Injection level output is made more amenable to manipulation according to context through introduction of a contrastive loss L_O, introducing a bias towards orthogonalization across context.

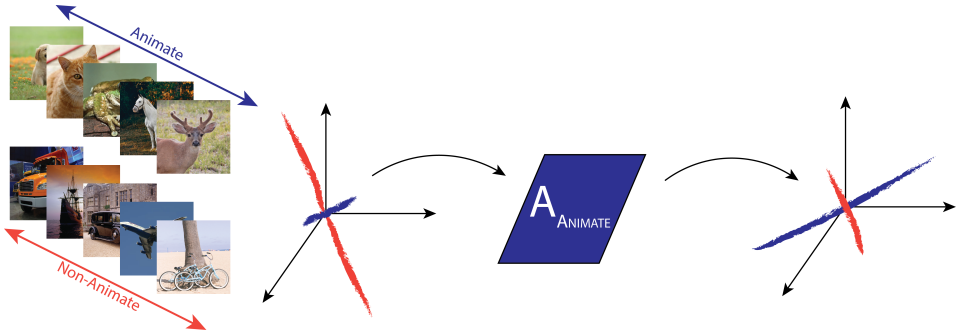


Figure 3: Illustration of the effect of affine application over injection level features. Shown are images from two contexts, and the injection level feature representations for those contexts produced by the injection level. If the dimension of the feature map of the injection level is HxWxC (Height x Width x Channel), this vector space is C-dimensional. The feature vectors are color coded according to context. After feature vector modulation through application of an affine, the characteristics of the context associated with that affine are more prominent. For example, with reference to Figure 1, if it is known that the context is one involving animals, rather than food, then characteristics may be biased towards fur interpretations, rather than kiwi fuzz or pastry surfaces.

Affine transformations are applied over the features of the injection level for the purpose of amplifying or dampening certain characteristics. This process aligns mid-level representations towards

conformance with higher level context expectation. The affines are made more effective through the disentanglement of characteristics at the injection level produced by the orthogonalizing bias.

During test time the CNN runs as a single stream (without the connection across the streams of multiple samples which the contrastive loss introduces), and a context expectation selects the affine to apply over the feature vectors of the injection level. This expectation can come from any source - in Section 4 we compare performance across network produced context expectations and ground truth context expectations.

3.1 Loss and Training

Training is broken into two stages, as detailed in Table 1. In the first, the base network is trained on its own, and features are biased towards orthogonality at the injection levels. In the second stage, the learning rate for the network parameters is reduced, and affines are added to the injection levels according to the context categories of input samples, initialized to identity matrices with added random noise, and given their own optimizers and learning rate. In the first stage, gradients backpropogate through the base network only, bypassing the affines; in the second stage gradients pass through both the base network and the affines. In each stage we employ batches containing equal proportions of samples belonging to each context.

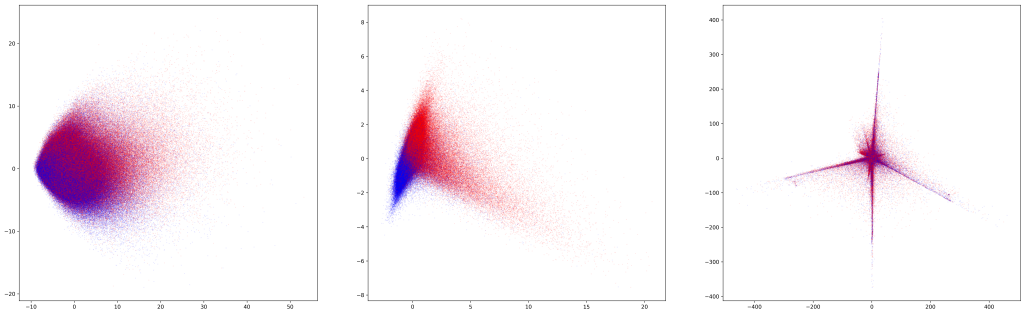
Training is broken into two stages for a few reasons: 1) this division allows the possibility of using pretrained networks and training affines in injection levels with no modification to the base network ($\lambda = 0$ and $LR_{N_2} = 0$), 2) allows fine-tuning of pretrained networks ($\lambda > 0$ and/or $LR_{N_2} > 0$), 3) we find that starting training of affines after the feature representations have had a chance to converge helps affine training - intuitively, the affines have to adapt to less of a moving target. We allow the network parameters to continue to train after affines have been introduced - at a reduced learning rate - as we find that this benefits performance.

To illustrate this process, consider a batch containing a *horse* and a *desk* image, *horse* belongs to animate while *desk* belongs to inanimate context. Both the images are fed in the same batch to the network during training. When training is in Stage 1, each image passes through the base network, sans affines. The feature representations of the injection levels for horse and desk are then connected to each other via L_O, and biased towards orthogonality with respect to each other. When training progresses to Stage 2, the L_O contrastive connection is removed, and affines are inserted according to context expectation. Both network and affine parameters are then updated according to gradients that pass through both affine and network parameters.

We find that the angles between mid-level features associated with higher level contexts can be increased to a much greater degree than they would be without this orthogonalizing loss, as measured by cosine similarity, without appreciable negative impacts on performance. See Figure 5 for an illustration of the extent to which this cosine loss is reduced when introducing this orthogonalizing loss.

	Stage 1	Stage 2
Loss	$L_1(Y, P, F) = \lambda * L_O(F, Y) + CE(Y, P)$	$L_2(Y, P, F) = CE(Y, P)$
Optimizers	$O_N(LR_{N_1}, L_1)$	$O_N(LR_{N_2}, L_2) + \sum_{i=0}^{\lvert C \rvert} O_{A_{c_i}}(LR_C, L_2)$

Table 1: Definition of losses, and their use in optimizers across Stages 1 and 2 of training. Y, P, and F correspond to sample labels, predictions, and injection level features, respectively. C is the context set. $L_O(F, Y)$ is the orthogonalization bias loss described in Equation 1, λ (intermediate loss scaling) is a scaling term, and $CE(Y, P)$ is cross entropy loss. $O_N(LR, L)$ is an optimizer applied over only the base network parameters, and $O_{A_{c_i}}$ is an optimizer applied over only the parameters of the affine associated with context c_i, for network learning rates LR_{N_1}, LR_{N_2}, affine learning rate LR_C, and losses L_1, L_2.



(a) PCA Projection, $\lambda = 0$ (b) PCA Projection, $\lambda = 2$ (c) PCA Projection, $\lambda = 50$

Figure 4: Illustrations of the effect of different levels of λ, the scaling term applied to orthogonalization bias L_O, applied to an injection level of the second to last conv op of the architecture given in the Appendix. Feature vectors of size 64 for the animate / inanimate split (see Appendix) of CIFAR100 or projected into 2 dimensions through Principal Component Analysis (PCA). Blue points correspond to features produced from animate images, and red points correspond to features produced by inanimate classes. Subfigure a illustrates context distribution when no orthogonalizing bias is used ($\lambda = 0$) - there is some degree of context separation, but it is not significant. A moderate degree of orthogonalizing bias ($\lambda = 2$) in subfigure 2 separates animate from inanimate features in the injection level. A significant degree of orthogonalizing bias ($\lambda = 50$) results in poor separation and degraded performance. See Appendix for more details, including UMAP projections.

$$L_O(F, Y) = \frac{1}{|S_{F,Y}|} \sum_{(f_1, f_2) \in S_{F,Y}} max(0, \frac{f_1 \cdot f_2}{\|f_1\| \|f_2\|}) \tag{1}$$

Where

$$S_{F,Y} = \{(f_1, f_2) | f_i \sim U(F_{c_i}), Y_C(f_1) \neq Y_C(f_2), IL(f_1) = IL(f_2)\} \tag{2}$$

Where

$$F_{c_i} = \{f | Y_C(f) = c_i\} \tag{3}$$

Where $|S_{F,Y}|$ is a method hyper-parameter, $Y_C(f)$ is the context of the sample from which feature vector f was produced, $IL(f)$ is the injection level from which f was taken, and $U(A)$ is the uniform probability distribution over elements of A.

With L_O we wish to separate the angles of features associated with different contexts, in order to better enable manipulation through affines. This can be seen as an exacerbation of CNN's natural tendency towards decoupled representations - where feature type has a tendency to group according to feature vector angle - through a structuring of characteristics' feature vector angles according to higher level context.

We do this through a cosine loss ($\frac{A \cdot B}{\|A\| \|B\|}$) applied to the features of the injection level. As we wish to control the network representation in terms of context expectation, we apply this loss across context. Figure 4 illustrates the behavior produced by the orthogonalizing bias.

4 Experiments

Here we cover experiments demonstrating the utility of our feedback method. We perform evaluations over CIFAR100 (Krizhevsky et al., 2009), ImageNet (Deng et al., 2009), and the Caltech

	CIFAR					ImageNet			Birds		
	1	2	3	4	Full	1	2	3	1	2	3
VGG	91.65 %	95.34	94.62%	95.11%	96.43%	98%	97.9%	97.8%	81.19%	90.3%	75.3%
ViT	90.32%	96.17%	93.46%	94.17%	96.2%	98.5%	94.3%	97%	80.4%	87.19%	80.29%
CNN	82.9%	85.5%	88.24%	85.5%	87.25%	89.33%	90%	92.46%	x	x	x

Table 2: Context classification accuracies across model and dataset splits. The simple CNN was not applied over Birds data as its expected input resolution is small.

	CIFAR					ImageNet			Birds		
	1	2	3	4	Full	1	2	3	1	2	3
VGG	87.5%	75.5%	87.1%	88.7%	72.64%	88.51%	86.5%	88.2%	70.37%	71.02%	71.80%
VGG (Pred Feedback)	**89.1%**	**76.8%**	**88.9%**	**89%**	**74.32%**	**90%**	**88.17%**	**89.1%**	**71.25%**	**76.6%**	**74.7**
ViT	81.77%	70.8%	75.32%	81.1%	71.1%	73.66%	76.03%	79.92%	70.25%	**69.5%**	71%
ViT (Pred Feedback)	**83.17%**	**76.97%**	**78.0%**	**81.9%**	**74.29%**	**74.0%**	**79.75%**	**80.24%**	**70.3%**	69.1%	**71.2%**
CNN	71.6%	57.1%	70.34%	71.3%	52%	57.8%	70.23%	64.8%	x	x	x
CNN (Pred Feedback)	**74.1%**	**59.5%**	**71.91%**	**74%**	**59.8%**	**58.06%**	**70.6%**	**66%**	x	x	x

Table 3: Object classification accuracies across model and dataset splits. Comparison between base VGG and feedback based on predicted context (Pred Feedback), where the context fed down the network is first predicted by the same network (accuracies for context prediction are given in Table 2). The simple CNN was not applied over Birds data as its expected input resolution is small.

UCSD Birds Data set (Birds) (Wah et al., 2011), all with multiple context splits - these datasets are described in Section B of the Appendix. We evaluate our method using both ground truth context expectations (**GT Feedback**), as well as context expectations derived from a network of the same structure as the base network (**Pred Feedback**). All experiments are conducted using a 6-layer CNN base architecture, a VGG-16 network (Simonyan & Zisserman, 2014), and a Transformer model (Mehta & Rastegari, 2021). The hyperparameters to the modifications made to each of the base architectures for feedback incorporation are described briefly in Sections 4.1 and 4.1.1, and described in detail in the Appendix. We show confusion matrices for a 10-class context split over CIFAR100 for both ground truth context expectations and network predicted context expectations, in Figures 6a and 6b, respectively.

4.1 MODELS

Here we describe the architectures in which we incorporate feedback. Each model consumes 1 GPU during train and test time. For all feedback experiments we choose a λ (intermediate loss scaling) of 1.0 (otherwise set to 0.0).

	CIFAR					ImageNet			Birds		
	1	2	3	4	Full	1	2	3	1	2	3
VGG	87.5%	75.5%	87.1%	88.7%	72.64%	88.51%	86.5%	88.2%	70.37%	71.02%	71.80%
VGG (GT Masking)	92.8%	81.2%	**89.8%**	93.4%	**74.2%**	89.6%	**89.4%**	90.44%	75.32%	77.26%	74.82%
VGG (GT Feedback)	**93.4%**	**81.4%**	88.8%	**93.8%**	74%	**89.8%**	89.07%	**91.36%**	**78.77%**	**79.23%**	**76.17%**
ViT	81.77%	70.8%	75.32%	81.1%	71.1%	73.66%	76.03%	79.92%	70.25%	69.5%	71%
ViT (GT Masking)	89.8%	78.1%	**79.3%**	87.8%	73%	76.5%	**79.81%**	**83.91%**	-	-	-
ViT (GT Feedback)	**90.22%**	**78.4%**	79.2%	**89.9%**	**73.1%**	**76.6%**	77.66%	80%	-	-	-
CNN	71.6%	57.1%	70.34%	74%	52%	57.8%	70.23%	64.8%	x	x	x
CNN (GT Masking)	81.3%	64.4%	71.37%	79.5%	55.7%	61.95%	71.45%	**72.3%**	x	x	x
CNN (GT Feedback)	**83.2%**	**64.4%**	**71.9%**	**80.7%**	**62.1%**	**62.85%**	**73%**	71.2%	x	x	x

Table 4: Object classification accuracies across model and dataset splits. Comparison between base VGG, masking based on Ground Truth (GT) context, and feedback based on ground truth context (GT Feedback). The simple CNN was not applied over Birds data as its expected input resolution is small. (Entries with "-" to be updated when complete).

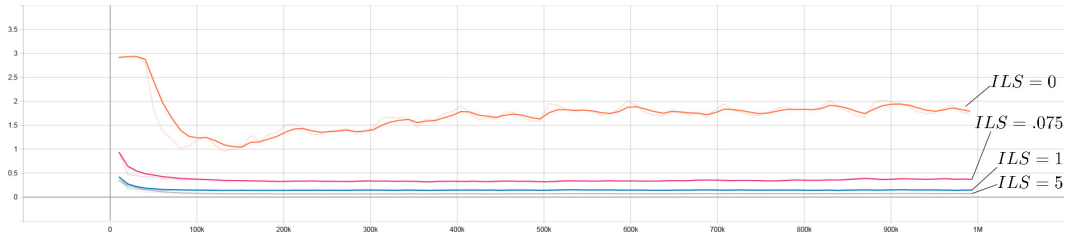


Figure 5: Shown above is a convergence plot of orthogonality loss L_O over several intermediate loss scaling values of $\{0, 0.075, 1, 5\}$. We employ a value of 0.075 in our experiments.

4.1.1 SHALLOW CNN

This model comprises a 6-layer CNN architecture, shown in Appendix, consisting of 3 by 3 shaped kernels, max-pooling applied over every other layer, and dropout ($p = 0.375$, $p = 0.1$) applied over the penultimate fully connected layer and after each convolution operation, respectively. The affine is applied after the second to last convolution operation, though we observe high performance inserting the affines anywhere throughout the second half of the architecture. We train the first stage for roughly 5 million iterations for all splits. A learning rate of 0.001 is chosen for the training of the base network during the first stage, and a learning rate of $5 * 10^{-5}$ is chosen for the learning rate of the base network during the second stage, whereas the affine learning rate is set to $1 * 10^{-3}$.

4.1.2 VGG

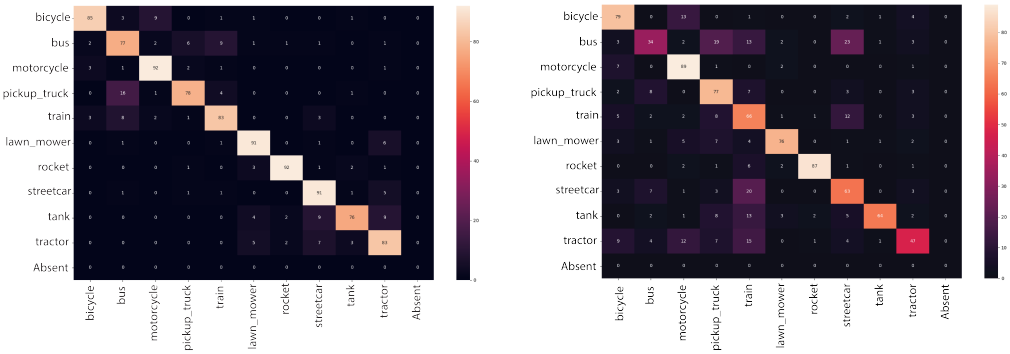
Here we adopt a VGG-16 network with pre-trained weights over ImageNet. The VGG network consists of 16 layers consisting of convolution and max-pooling operations. The affine is applied after the eleventh convolution operation, though we observe high performance inserting the affines anywhere throughout the last six layers. We train the first stage for roughly 1.5 million iterations for all splits, until smooth convergence. A learning rate of $5 * 10^{-6}$ is chosen for the training of the base network during the first stage, and a learning rate of $2.5 * 10^{-6}$ is chosen for the learning rate of the base network during the second stage, whereas the affine learning rate is set to $2.5 * 10^{-3}$.

4.1.3 VISUAL TRANSFORMER

Here we adopt a variant of the Visual Transformer models (Mehta & Rastegari, 2021), a general-purpose vision transformer that outperforms many related visual transformer architectures while being lightweight and easy to train. The affine is applied immediately after the third last attention block. We train the first stage for roughly 2.0 million iterations for all splits, until smooth convergence. A learning rate of $1 * 10^{-3}$ is chosen for the training of the base network during the first stage, and a learning rate of $1.0 * 10^{-6}$ is chosen for the learning rate of the base network during the second stage, whereas the affine learning rate is set to $1 * 10^{-3}$.

4.2 ORACLE EVALUATION

The ground truth context model (**GT Feedback**) assumes access to the ground truth context belonging to each input sample during training and test time, using contextual knowledge to index into the affine transforms for application over mid-level features as described in Section 3. Here we evaluate the extent to which our complete framework outperforms 1) a base CNN architecture mirroring that of our framework, and 2) the same base architecture where a hard masking operation using the ground truth contextual knowledge is applied over its class-level outputs. The **GT Masking** baseline corresponds to the same base CNN architecture where a hard mask of $c_g \in \{0, 1\}^k$ is applied over the output of the network, where k corresponds to the number of classes and c_g takes on a value of 1 for within-context classes and 0 for out-of-context classes.



(a) With orthogonalization bias and affines.

(b) With neither orthogonalization bias nor affines.

Figure 6: Confusion matrices for a our simple architecture (see Appendix) with and without feedback (orthogonalization bias and with application of affines) - subfigure a and b respectively - over the vehicles 1 vs vehicles 2 split of CIFAR100. The first 5 rows correspond to the first context, and the next 5 rows correspond to the second context. Note that cross context confusion (quadrants 1 and 3 of the confusion matrices) is significantly reduced with feedback.

4.3 Contextual Prediction

Context prediction is performed through the addition of a second logits classification head to the base networks. This head is trained in conjunction with the object classification head using ground truth context labels.

4.4 Discussion

In Table 3 when ground truth context is not known, we observe that models with feedback using context predicted by the same model (**Pred Feedback**) consistently outperform models which do not use feedback. Further, in Table 4 we observe superior performance for most runs of **GT Feedback** over both non-feedback and **GT Masking** runs.

5 Conclusion

We have presented an argument for the utility of feedback in vision. Feedback is 1) prominent in biological vision, with the majority of neural connections in the cortex consisting of feedback connections, 2) allows better constraining of under-constrained processes of abstraction, 3) allows for the online adaptation of vision systems towards alignment with high level understanding of the world. We leverage the fact that CNNs have a tendency towards decoupled representations, exacerbating the separation of mid-level features associated with different higher level contexts. This allows better direct manipulation of the level at which feedback is introduced, minimizing collateral effects on characteristics not being selected for. In contrast to post-hoc filtering of interpretations for consistency with context expectations, MVF allows for cross-context detections and produces higher accuracies. MVF involves a top-down bridging of the signal-symbol gap, making it applicable to systems ranging from image and video understanding to explainable AI and robotics. In the future this work will be extended to localization, e.g. object detection or semantic segmentation, as well as used in embodied contexts with an active agent.

References

Ruzena Bajcsy. Active perception. *Proceedings of the IEEE*, 76(8):966–1005, 1988.

Chunshui Cao, Xianming Liu, Yi Yang, Yinan Yu, Jiang Wang, Zilei Wang, Yongzhen Huang, Liang Wang, Chang Huang, Wei 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*, pp. 2956–2964, 2015.

Chunshui Cao, Yongzhen Huang, Yi Yang, Liang Wang, Zilei Wang, and Tieniu Tan. Feedback convolutional neural network for visual localization and segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 41(7):1627–1640, 2018.

Isaac Caswell, Chuanqi Shen, and Lisa Wang. Loopy neural nets: Imitating feedback loops in the human brain. *Tech. Report*, 2016.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255. Ieee, 2009.

Charles D Gilbert and Wu Li. Top-down influences on visual processing. *Nature Reviews Neuroscience*, 14(5):350–363, 2013.

Charles D. Gilbert and Mariano Sigman. Brain states: Top-down influences in sensory processing. *Neuron*, 54(5):677–696, 2007. ISSN 0896-6273. doi: https://doi.org/10.1016/j.neuron.2007.05.019. URL https://www.sciencedirect.com/science/article/pii/S0896627307003765.

P. Hu and D. Ramanan. Bottom-up and top-down reasoning with hierarchical rectified gaussians. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5600–5609, Los Alamitos, CA, USA, jun 2016. IEEE Computer Society. doi: 10.1109/CVPR.2016.604. URL https://doi.ieeecomputersociety.org/10.1109/CVPR.2016.604.

Gabriel Kreiman and Thomas Serre. Beyond the feedforward sweep: feedback computations in the visual cortex. *Annals of the New York Academy of Sciences*, 1464(1):222–241, 2020.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

Kestutis Kveraga, Avniel S Ghuman, and Moshe Bar. Top-down predictions in the cognitive brain. *Brain and cognition*, 65(2):145–168, 2007.

Qianli Liao and Tomaso A. Poggio. Bridging the gaps between residual learning, recurrent neural networks and visual cortex. *CoRR*, abs/1604.03640, 2016. URL http://arxiv.org/abs/1604.03640.

Drew Linsley, Alekh Karkada Ashok, Lakshmi Narasimhan Govindarajan, Rex Liu, and Thomas Serre. Stable and expressive recurrent vision models. *Advances in Neural Information Processing Systems*, 33:10456–10467, 2020a.

Drew Linsley, Junkyung Kim, Alekh Ashok, and Thomas Serre. Recurrent neural circuits for contour detection. *arXiv preprint arXiv:2010.15314*, 2020b.

Weiyang Liu, Zhen Liu, Zhiding Yu, Bo Dai, Rongmei Lin, Yisen Wang, James M. Rehg, and Le Song. Decoupled networks. *CoRR*, abs/1804.08071, 2018. URL http://arxiv.org/abs/1804.08071.

NV Kartheek Medathati, Heiko Neumann, Guillaume S Masson, and Pierre Kornprobst. Bio-inspired computer vision: Towards a synergistic approach of artificial and biological vision. *computer vision and image understanding*, 150:1–30, 2016.

Sachin Mehta and Mohammad Rastegari. Mobilevit: light-weight, general-purpose, and mobile-friendly vision transformer. *arXiv preprint arXiv:2110.02178*, 2021.

Sofia Paneri and Georgia G. Gregoriou. Top-down control of visual attention by the prefrontal cortex. functional specialization and long-range interactions. *Frontiers in Neuroscience*, 11, 2017. ISSN 1662-453X. doi: 10.3389/fnins.2017.00545. URL https://www.frontiersin.org/article/10.3389/fnins.2017.00545.

Pedro Pinheiro and Ronan Collobert. Recurrent convolutional neural networks for scene labeling. In Eric P. Xing and Tony Jebara (eds.), *Proceedings of the 31st International Conference on Machine Learning*, volume 32 of *Proceedings of Machine Learning Research*, pp. 82–90, Bejing, China, 22–24 Jun 2014. PMLR. URL https://proceedings.mlr.press/v32/pinheiro14.html.

Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE international conference on computer vision*, pp. 618–626, 2017.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

Ruben S van Bergen and Nikolaus Kriegeskorte. Going in circles is the way forward: the role of recurrence in visual inference. *Current Opinion in Neurobiology*, 65:176–193, 2020.

Catherine Wah, Steve Branson, Peter Welinder, Pietro Perona, and Serge Belongie. The caltech-ucsd birds-200-2011 dataset. 2011.

Jian Yao, Sanja Fidler, and Raquel Urtasun. Describing the scene as a whole: Joint object detection, scene classification and semantic segmentation. In *2012 IEEE conference on computer vision and pattern recognition*, pp. 702–709. IEEE, 2012.

A. R. Zamir, T.-L. Wu, L. Sun, W. Shen, B. E. Shi, J. Malik, and S. Savarese. Feedback Networks. *ArXiv e-prints*, December 2016.

A Appendix

Abstract

We here present Appendix. Section B describes datasets; Section C presents the base CNN architecture used in experiments; Section E details tuned parameters; and, Section F presents data splits.

B Datasets

B.1 CIFAR100

We adopt CIFAR100 for the 1) high-levels of visual ambiguity due to low resolution, 2) existence of several distinct "superclasses" consisting of a roughly equal number of classes, and 3) the cross-context confusion across classes highly similar in appearance (e.g., sharks and dolphins). We adopt the official training and test split for the CIFAR100 dataset. Each class contains exactly 500 training images and 100 testing images, with each superclass consisting of 5000 training images and 1000 testing images. We use the CIFAR100 superclasses in constructing context splits. Split 1: Vehicles 1 vs. Vehicles 2, Split 2: Household Devices vs. Furniture, Split 3: Aquatic Mammals vs. Fish. For full class breakdown see Appendix.

B.2 ImageNet

We adopt ImageNet for several of the aforementioned reasons above, as well as its generality in that it spans 1000 unique classes. Each class contains variable number of images - we designate 80% of the images in each class for training, but only 2% for testing due to the computational cost incurred by the high number of images and the need for frequent testing.

We employ the following context split over ImageNet, designed to be similar to CIFAR100 splits, the full class breakdown of which is given in Appendix: Split 1: Household Devices vs. Furniture, Split 2: Aquatic Mammals vs. Fish, Split 3: Vehicles 1 vs. Vehicles 2.

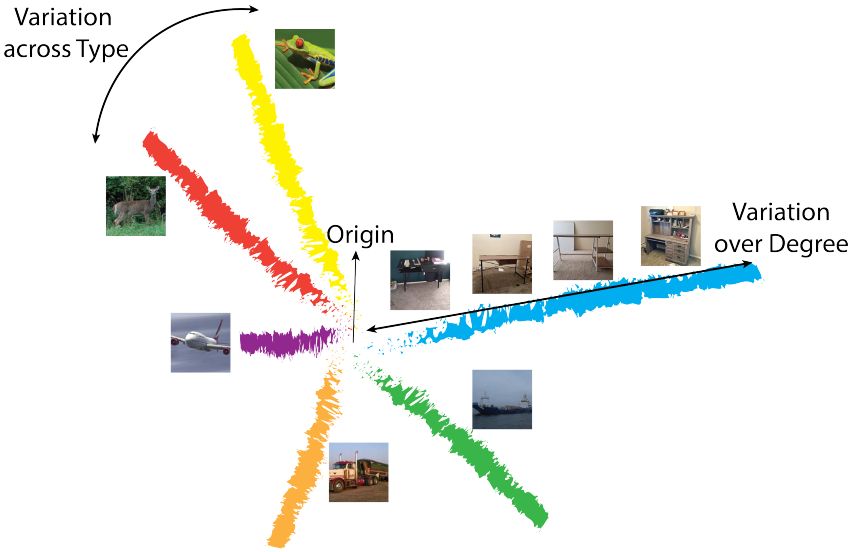


Figure 7: Illustration of decoupled representations, as may be learned by a CNN. For illustration we annotate characteristics with visually recognizable categories, though in this work we exploit the tendency towards decoupled representations at lower levels. Feature vector angle corresponds to characteristic type, while feature vector magnitude corresponds to within characteristic variation or degree. (Liu et al., 2018) observe that CNNs produce decoupled representations, and derive a similar illustration over MNIST by setting a convolution operator's dimension to 2.

C BASE CNN

Figure 8 illustrates the base CNN model (apart from VGG and ViT), for which performance is reported in Tables in the main paper.

D DECOUPLED REPRESENTATIONS

CNNs have a natural tendency towards *decoupled* representations. These are representations where characteristics have a tendency to be represented in such a way that feature vector angle corresponds to characteristic type, while feature vector magnitude corresponds to characteristic variation or degree (Liu et al., 2018).

An illustration of this is given in Figure 7.

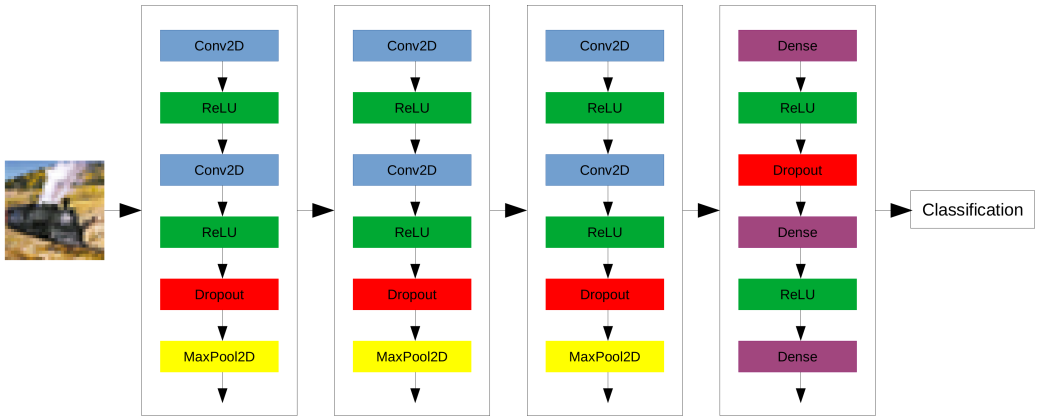


Figure 8: Illustration of the base architecture used in these experiments. The injection level used in these experiments is after the ReLU of the second to last Conv2D operator.

E PARAMETERS

We here list parameters' tuned values not introduced in the main paper:

1. Image size: 32×32 for CIFAR100, 224×224 for ImageNet.
2. Model input image size: 32×32 for 6-layer CNN, 224×224 for VGG16. Images resized using bilinear interpolation.
3. Size of feature set selected for orthogonalization: 25.
4. Batch size: 256 (CIFAR100 splits), 64 (ImageNet splits).
5. Data augmentations: Random rotations (15 degrees), random resized crops, Random horizontal flips.
6. Feedback Base Model: ADAM's optimizer, weight decay of 7.5×10^{-4} for both stages and both models.
7. Affine Optimizer: ADAM's optimizer, affine learning rate of 0.001 for second stage training of both models.
8. Context Model: ResNet18 model with pretrained weights over ImageNet and learning rate of 0.001 using SGD optimizer.

F DATA SPLITS

F.1 CUB-200-2011

We adopt the Caltech-UCSD-Birds dataset for several of the aformentioned reasons above, in particular for the high cross-context confusion across different species of birds highly similar in appearance. It consists of 11,788 images with 200 classes corresponding to bird species. Like the Imagenet dataset, we designate 80% of the dataset for training and 20% for testing.

We employ the following splits over the CUB dataset: Splits based on migration behavior, splits based on trophic level () and splits based on primary lifestyle.

F.2 SPLITCIFAR

CIFAR100 Dataset Sub-Splits

Split	Group	Classes
1	Vehicles 1 Vehicles 2	**Bicycle, Bus, Motorcycle, Pickup truck** **Lawn mower, Rocket, Streetcar, Tank, Tractor**
2	Household Devices Furniture	**Clock, Keyboard, Lamp, Telephone, television** **Bed, Chair, Couch, Table, Wardrobe**
3	Aquatic mammals Fish	**Beaver, Dolphin, Otter, Seal, Whale** **aquarium fish, flatfish, ray, shark, trout**
4	Small animals Large animals	**fox, porcupine, possum, raccoon, skunk** **bear, leopard, lion, tiger, wolf**

Full CIFAR100 Split:

animate = beaver, dolphin, otter, seal, whale, aquarium_fish, flatfish, ray, shark, trout, bear, leopard, lion, tiger, wolf, camel, cattle, chimpanzee, elephant, kangaroo, fox, porcupine, possum, raccoon, skunk, baby, boy, girl, man, woman, crocodile, dinosaur, lizard, snake, turtle, hamster, mouse, rabbit, shrew, squirrel, bee, beetle, butterfly, caterpillar, cockroach, crab, lobster, snail, spider, worm

inanimate = orchid, poppy, rose, sunflower, tulip, bottle, bowl, can, cup, plate, apple, mushroom, orange, pear, sweet_pepper, clock, keyboard, lamp, telephone, television, bed, chair, couch, table, wardrobe, bridge, castle, house, road, skyscraper, cloud, forest, mountain, plain, sea, maple_tree, oak_tree, palm_tree, pine_tree, willow_tree, bicycle, bus, motorcycle, pickup_truck, train, lawn_mower, rocket, streetcar, tank, tractor

F.3 SPLITIMAGENET

ImageNet Dataset Splits

Split	Group	Classes
1	Household Devices Furniture	**analog clock, digital clock, wall clock, computer keyboard, dial telephone, table lamp, television, cellular telephone** **studio couch, dining table, wardrobe, folding chair**
2	Aquatic mammals Fish	**Beaver, Dolphin, Otter, Seal, Whale** **barracouta, eel, coho, rock beauty, anemone fish, sturgeon, gar, puffer, lionfish**
3	Devices 1 Devices 2	**mountain bike, bicycle-built-for-two,school bus, moped, tricycle, bullet train, passenger car, pickup** **lawn mower, tractor, streetcar, tank**