# REVERSIBLE COLUMN NETWORKS

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

We propose a new neural network design paradigm *Reversible Column Networks (RevCols)*. The main body of RevCols is composed of multiple copies of subnetworks, named *columns* respectively, between which multi-level reversible connections are employed. Such architectural scheme attributes RevCols very different behavior from conventional networks: during forward propagation, features in RevCols are learned to be gradually *disentangled* when passing through each column, whose total information is maintained rather than *compressed* or discarded as other network does. Our experiments suggest that CNN-style RevCols can achieve very competitive performances on multiple computer vision tasks such as image classification, object detection and semantic segmentation, especially with large parameter budget and large dataset. For example, after ImageNet-22K pre-training, RevCol-XL obtains 88.2% ImageNet-1K accuracy. Given more pre-training data, our largest model RevCol-H reaches **90.0%** on ImageNet-1K, **61.2%** $\text{AP}_{box}$ and **53.6%** $\text{AP}_{mask}$ on COCO detection test-dev set, **57.1%** mIoU on ADE20k segmentation. To our knowledge, it is the best COCO detection result among *pure* CNN models without extra detection data. Moreover, as a general macro architecture fashion, RevCols can also be introduced into transformers or other neural networks, which is demonstrated to improve the performances in both computer vision and NLP tasks.

## 1 INTRODUCTIONS

*Information Bottleneck principle (IB)* (Tishby et al., 2000; Tishby & Zaslavsky, 2015) rules the deep learning world. Consider a typical supervised learning network as in Fig. 1 (a): layers close to the input contain more low-level information, while features close to the output are rich in semantic meanings. In other words, information unrelated to the target is gradually *compressed* during the layer-by-layer propagation. Although such learning paradigm achieves great success in many practical applications, it might not be the optimal choice in the view of *feature learning* – down-stream tasks may suffer from inferior performances if the learned features are *over* compressed, or the learned semantic information is irrelevant to the target tasks, especially if a significant domain gap exists between the source and the target tasks (Zamir et al., 2018). Researches have devoted great efforts to make the learned features to be more universally applicable, *e.g.* via *self-supervised pre-training* (Oord et al., 2018; Devlin et al., 2018; He et al., 2022; Xie et al., 2022) or multi-task learning (Ruder, 2017; Caruana, 1997; Sener & Koltun, 2018).

In this paper, we mainly focus on an alternative approach: building a network to learn *disentangled representations*. Unlike *IB* learning, disentangled feature learning (Desjardins et al., 2012; Bengio et al., 2013; Hinton, 2021) does not intend to extract the most related information while discard the less related; instead, it aims to embed the task-relevant concepts or semantic words into a few decoupled dimensions respectively. Meanwhile the whole feature vector roughly maintains as much information as the input. It is quite analogous to the mechanism in biological cells (Hinton, 2021; Lillicrap et al., 2020) – each cell shares an identical copy of the whole genome but has different expression intensities. Accordingly in computer vision tasks, learning disentangled features is also reasonable: for instance, high-level semantic representations are tuned during *ImageNet* pre-training, meanwhile the low-level information (*e.g.* locations of the edges) should also be maintained in other feature dimensions in case of the demand of down-stream tasks like object detection.

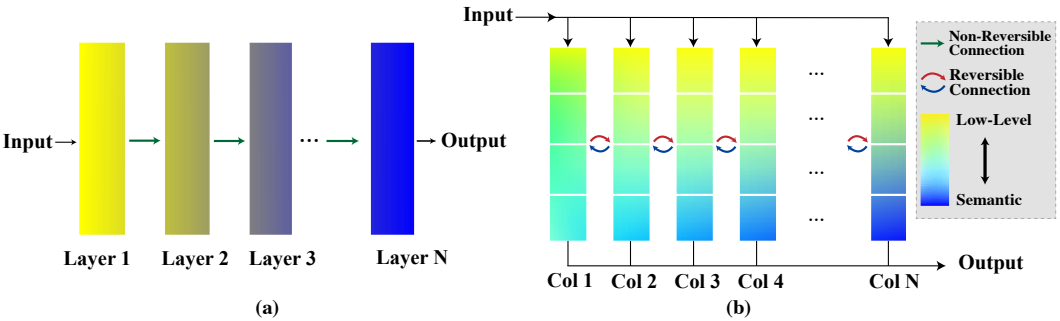


Figure 1: Sketch of the information propagation in: (a) Vanilla single-column network. (b) Our reversible column network. Yellow color denotes low-level information and blue color denotes semantic information.

Fig. 1 (b) sketches our main idea: *Reversible Column Networks (RevCols)*, which is greatly inspired by the big picture of *GLOM* (Hinton, 2021). Our network is composed of $N$ subnetworks (named *columns*) of identical structure (however whose weights are not necessarily the same), each of which receives a copy of the input and generates a prediction. Hence multi-level embeddings, *i.e.* from low-level to highly semantic representations, are stored in each column. Moreover, *reversible transformations* are introduced to propagate the multi-level features from $i$-th column to $(i+1)$-th column without information loss. During the propagation, since the complexity and nonlinearity increases, the quality of all feature levels is expected to gradually improve. Hence the last column (Col $N$ in Fig. 1 (b)) predicts the final disentangled representations of the input.

In RevCols, one of our key contributions is the design of the *reversible transformations* between adjacent columns. The concept is borrowed from the family of *Reversible Networks* (Chang et al., 2018; Gomez et al., 2017; Jacobsen et al., 2018; Mangalam et al., 2022); however, conventional reversible structures such as *RevNets* (Gomez et al., 2017) (Fig. 2 (a)) usually have two drawbacks: first, feature maps within a reversible block are restricted to have the same shape*; second, the last two feature maps in RevNets have to contain both low-level and high-level information due to the reversible nature, which may be difficult to optimize as in conflict with *IB* principle. In this paper, we overcome the drawbacks by introducing a novel reversible multi-level fusion module. The details are discussed in Sec. 2.

We build a series of CNN-based *RevCols* under different complexity budgets and evaluate them in mainstream computer vision tasks, such as *ImageNet* classification, *COCO* object detection and instance segmentation, as well as *ADE20K* semantic segmentation. Our models achieve comparable or better results than sophisticated CNNs or *vision transformers* like ConvNeXt (Liu et al., 2022b) and Swin (Liu et al., 2021). For example, after ImageNet-22K pre-training, our RevCol-XL model obtains **88.2%** accuracy on ImageNet-1K without using transformers or large convolutional kernels (Ding et al., 2022b; Liu et al., 2022b; Han et al., 2021). More importantly, we find RevCols can scale up well to large models and large datasets. Given a larger private pre-training dataset, our biggest model RevCol-H obtains **90.0%** accuracy on ImageNet-1K classification, **61.2%** $AP_{box}$ and **53.6%** $AP_{mask}$ on COCO detection test-dev set, and **57.1%** mIoU on ADE20K segmentation, respectively, without using extra downstream task data (*e.g. Objects365* (Shao et al., 2019)) for pre-training. To our knowledge, it is the best reversible model on those tasks, as well as the best *pure* CNN model on COCO. In the appendix, we further demonstrate RevCols can work with transformers (Dosovitskiy et al., 2020; Devlin et al., 2018) and get improved results on both computer vision and NLP tasks. Finally, similar to RevNets (Gomez et al., 2017), RevCols also shares the bonus of memory saving from the reversible nature, which is particularly important for large model training.

**Relation to previous works.** Although our initial idea on feature disentangling is derived from *GLOM* (Hinton, 2021), in *RevCols* there are a lot of simplifications and modifications. For example, GLOM does not involve reversible transformations between columns; it suggests contrastive auxiliary loss to avoid feature collapse. As for other multi-scale grid-like architectures such as *HRNets* (Wang et al., 2020), *DEQ models* (Bai et al., 2020) and *FPNs* (Lin et al., 2017; Tan et al., 2020), the design

*In precise, feature maps of odd and even indexes should be equal sized respectively.

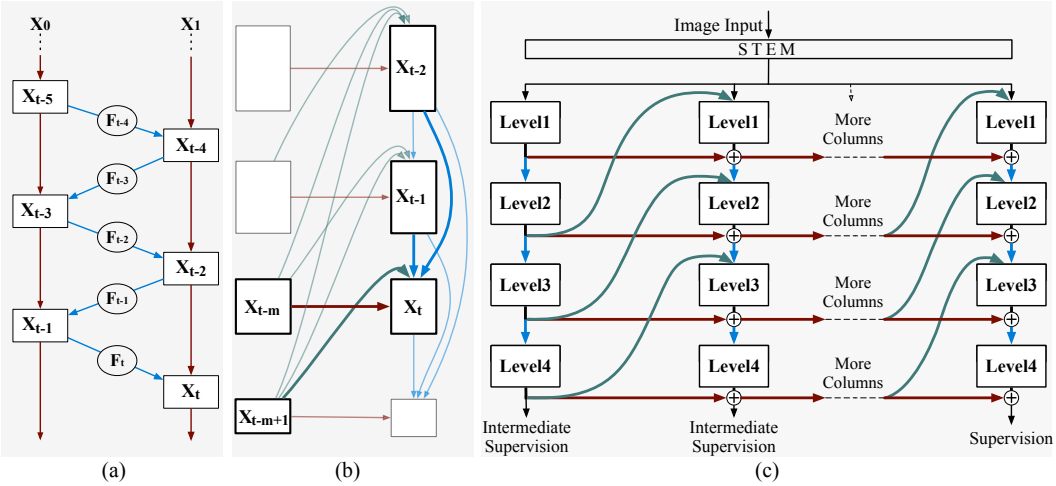


Figure 2: (a) Reversible unit in RevNet (Gomez et al., 2017). (b) Multi-level reversible unit. All inputs for level $t$ are highlighted. (c) An overview of the whole reversible column network architecture, with simplified multi-level reversible unit.

purpose of those models is to fuse multi-scale features rather than learn disentangled representations; therefore, in general they still follow the paradigm in Fig. 1 (a) – neither multiple entrances/exits nor reversible structures are employed. Very recently, *RevBiFPN* (Chiley et al., 2022) comes up with an reversible variant of FPN, which is further employed in an HRNet-like architecture. Though our RevCols shares the similar idea of multi-scale reversible transformations with RevBiFPN, our work is done independently, which is derived from a different motivation of feature disentangling, and has much simpler architectures (*e.g.* free of reversible upsampling tower) and higher performances. We compare some of those models in Sec. 3.

## 2 METHOD

In this section, we introduce the design details of our *Reversible Column Networks (RevCols)*. Fig. 1 (b) illustrates the top-level architecture. Notice that for each column in RevCols, for simplicity we directly reuse existing structures such as *ConvNeXt* (Liu et al., 2022b), hence in the following subsections, we mainly focus on how to build the reversible connections between columns. In addition, we introduce an plug-and-play intermediate supervision on top of each column, which further improves the training convergence and feature quality.

### 2.1 MULTI-LEVEL REVERSIBLE UNIT

In our network, *reversible transformations* plays a key role in feature disentangling without information loss, whose insight comes from *Reversible Neural Networks* (Chang et al., 2018; Gomez et al., 2017; Jacobsen et al., 2018; Mangalam et al., 2022). Among them, we first take a review of one representative work *RevNet* (Gomez et al., 2017). As shown in Fig. 2 (a), RevNet first partitions the input $x$ into two groups, $x_0$ and $x_1$. Then for later blocks, for example, block $t$, it takes two anterior blocks' outputs $x_{t-1}$ and $x_{t-2}$ as input and generates the output $x_t$. The mapping of block $t$ is *reversible*, i.e. $x_{t-2}$ can be reconstructed by two posterior blocks $x_{t-1}$ and $x_t$. Formally, the forward and backward[†] computation follow the equations:

$$\begin{aligned} Forward: x_t &= \boldsymbol{F}_t(x_{t-1}) + \gamma x_{t-2} \\ Backward: x_{t-2} &= \gamma^{-1}[x_t - \boldsymbol{F}_t(x_{t-1})], \end{aligned} \tag{1}$$

[†]Following the notation in Gomez et al. (2017), here "backward" means the inverse of forward formulation rather than backward propagation (gradient computing) in network training. Nonetheless, of course the backward formulation can be employed in computing gradient to reduce memory cost.

where $\boldsymbol{F}_t$ denotes an arbitrary non-linear operation analogous to those residual functions in standard *ResNets*; $\gamma$ is a simple reversible operation (*e.g*. channel-wise scaling), whose inverse is denoted by $\gamma^{-1}$. As mentioned in the introduction, the above formulation involves too strong constraint on the feature dimensions, i.e. $x_t, x_{t+2}, x_{t+4}, ...$ have to be equal sized, which is not flexible in architecture design. That is why RevNets (Gomez et al., 2017) introduce some non-reversible down-sampling blocks between reversible units, hence the whole network is not fully reversible. More importantly, we find there is no clear way directly employing Eq. 1 to bridge the columns in Fig. 1 (b).

To address the issue, we generalize Eq. 1 into the following form:

$$\begin{aligned} Forward: x_t &= \boldsymbol{F}_t(x_{t-1}, x_{t-2}, ..., x_{t-m+1}) + \gamma x_{t-m} \\ Backward: x_{t-m} &= \gamma^{-1}[x_t - \boldsymbol{F}_t(x_{t-1}, x_{t-2}, ..., x_{t-m+1})], \end{aligned} \tag{2}$$

where $m$ is the *order* of the recursion ($m \geq 2$). Clearly, the extension is still reversible. Then we partition every $m$ feature maps into a group: $(x_1, x_2, \ldots, x_m), (x_{m+1}, x_{m+2}, \ldots, x_{2m}), \ldots$. Given the features within any of the group, we can easily compute the features in other groups recursively according to Eq. 2. Compared with the original form, Eq. 2 has the following two nice properties:

- The constraint on the feature map sizes is greatly relaxed if $m$ is relatively large. Notice that Eq. 1 does not require feature maps *within* each group to be equal sized; such constraint only exist between groups. Therefore, we can use tensors of different shape to represent features of different semantic levels or different resolutions.
- Eq. 2 can easily cooperate with existing network architectures, even though the latter is not reversible. For example, we can assign $m$ feature maps in a standard *ResNet* to represent the feature maps within a group $(x_t, x_{t+1}, \ldots, x_{t+m-1})$, which is still compatible with Eq. 2 since ResNet can be viewed as a part of $(\boldsymbol{F}_t, \boldsymbol{F}_{t+1}, \ldots, \boldsymbol{F}_{t+m-1})$ respectively. Thus the whole network is still reversible.

Therefore, we can reorganize Eq. 2 into a *multi-column* fashion, as shown in Fig. 2 (b). Each column is composed of $m$ feature maps within a group, as well as their mother network. We name it *multi-level reversible unit*, which is the basic component of our RevCol as in Fig. 1 (b).

## 2.2 Reversible Column Architecture

### 2.2.1 Macro Design

An overview of our RevCol is presented in Fig. 2 (c). An input image is first split into non-overlapping patches by a patch embedding module. After that, patches are fed into every column. Same as other hierarchical networks, we adopt four levels in each column with gradually reduced feature maps. For classification tasks, we only use feature map of the last level (Level 4) in the last column for rich semantic information. For down-stream tasks, we use feature maps of all four levels in the last column as they contain both low-level and semantic information.

We simplify the multi-column reversible in Sec. 2.1. Rather than $m-1$ inputs for non-linear operation, we use only one low-level feature $x_{t-1}$ and one high-level feature $x_{t-m+1}$ as input. More inputs bring minor accuracy gain but consume much GPU resources. Eq. 2 is simplified as:

$$\begin{aligned} Forward: x_t &= \boldsymbol{F}_t(x_{t-1}, x_{t-m+1}) + \gamma x_{t-m} \\ Backward: x_{t-m} &= \gamma^{-1}[x_t - \boldsymbol{F}_t(x_{t-1}, x_{t-m+1})] \end{aligned} \tag{3}$$

In forward propagation, we backup the weights of each column and the features of the last column. In the backward propagation, we reconstruct the feature map $x_{t-m}$ first, then the forward equation can be recomputed with features $x_{t-1}, x_{t-m+1}$ and $x_{t-m}$ and backed-up weights to calculate the gradient.

In RevCol, information propagates in two directions, both inside column and inter-column. In the beginning, information is entangled in first column. As information iteratively propagates between columns, semantic information gradually sinks to bottom, since closing to output. Other low-level information is also maintained inside column, for the reversible property prevents the information from being discarded. In the end, low-level and semantic information appears in a disentangled manner in the last column. For detailed analysis, please refer to Appendix G.

In RevCol architecture, column number serves as a new dimension in addition to depth (number of blocks), and width (channels of each block) in vanilla single-column CNNs or ViTs. Increasing column numbers has similar income as increasing both width and depth in certain range.

#### 2.2.2 MICRO DESIGN

In this section, we provide the architecture design at a micro scale in RevCol. The micro design includes: 1) fusion module before each level's building blocks that fuses the input feature maps with different resolutions. 2) building blocks within each column. 3) reversible operation $\boldsymbol{\gamma}$.

**Fusion module** In each level of ConvNeXt, the inputs are first down-sampled in a patch-merging block. Then the outputs are passed through a bunch of residual blocks. In RevCol, as each level receives inputs of different resolutions, we introduce a fusion module which is composed of a down-sample block and an up-sample block to fuse the input first. For the down-sample block. we modify the original patch-merging block in ConvNeXt. We put the LayerNorm after the patch-merging convolution rather than before. Channel numbers are doubled in patch-merging convolution. The up-sample block is composed of a linear channel mapping layer, a LayerNorm normalization and a feature map interpolation layer. We halve the channel numbers in linear channel mapping layer. The outputs of down-sample block and up-sample block are element-wise summed, then passed to the residual blocks followed by.

**Building blocks within each column** We adopt the residual block proposed in ConvNeXt (Liu et al., 2022b), but revise the kernel size back to 3×3. On the one hand, we hope to keep more local low-level information in columns. On the other hand, information continually exchanges during column iterations, which allow our RevCol to achieve equivalent performance gain as models with large receptive field. Increasing kernel size only has minor accuracy gain, please refer to Appendix C.1 for more details.

**Reversible Operation $\boldsymbol{\gamma}$** We adopt a learnable reversible channel-wise scaling as reversible operation $\gamma$ to keep the network stable. Each time the features are summed up in forward of Eq. 3, the magnitude grows larger, which makes the training process unstable. Using a learnable scaling can suppress the magnitude of features. During training, we truncate the absolute value of learnable channel-wise scaling $\gamma$ less than $1e^{-3}$, because the numerical error could become large in the reverse computation when $\gamma$ is too small.

The design details can be found in Appendix A. We also generalize RevCol design to transformer, which is shown in Appendix B.

### 2.3 INTERMEDIATE SUPERVISION

Though multi-level reversible unit is able to maintain information during column iteration, the down-sample block still can discard information *inside* column. Features at the end of the front columns is too close to the final output, for reversible connections simply do scaling and summation. Such information loss leads to inferior performance. Similar problem also happens when using deeply-supervised method (Lee et al., 2015; Szegedy et al., 2015).

To mitigate the problem of information collapse, we propose an intermediate supervision method which adds additional supervision into front columns. For features in front columns, we hope to keep the mutual information between features and the input image as much as possible, so that the network discard less information within columns. Consider RevCol gradually disentangle semantic and low-level information, extracting and leveraging the task-relevant information can further boost the performance. Therefore, we need to maximize the lower bound of mutual information between features and the prediction.

Inspired by Wang et al. (2021), we add two auxiliary heads to features in the bottom level (Level 4). One is a decoder to reconstruct the input image, the other is a linear classifier. The decoder can be several blocks as He et al. (2022), or can be a light-weight single layer as Xie et al. (2022). We train the image reconstruction decoder using binary cross-entropy (BCE) loss. Compared with commonly used L1 and L2 loss, BCE loss output smoother value and is suitable to use along with CE Loss, since our training process is end-to-end. And the linear classifier can be trained in a regular classification fashion with the cross-entropy (CE) loss. For intermediate supervision at one column, the compound loss is the weighted summation of the above two loss. Note that intermediate supervision may not

Table 1: **ImageNet classification results.** We compare our models with state-of-the-art ○ Vision Transformers and ● CNNs that have comparable FLOPs and parameters. ↑ denotes models fine-tuning using image size larger than $224^2$. We report the top-1 accuracy on the validation set of ImageNet as well as the number of parameters and FLOPs. Our models are highlighted in gray.

| Model | Image Size | Params (M) | FLOPs (G) | Top-1 Acc. |
|---|---|---|---|---|
| *ImageNet-1K trained models* | | | | |
| ○ Swin-T (Liu et al.) | $224^2$ | 28 | 4.5 | 81.3 |
| ○ DeiT-S (Touvron et al. | $224^2$ | 22 | 4.6 | 79.8 |
| ○ Rev-ViT-S (Mangalam et al.) | $224^2$ | 22 | 4.6 | 79.9 |
| ● RevBiFPN-S3 (Chiley et al.) | $288^2$ | 20 | 3.3 | 81.1 |
| ● EfficientNet-B4 (Tan & Le) | $380^2$ | 19 | 4.2 | 82.9 |
| ● ConvNeXt-T (Liu et al.) | $224^2$ | 29 | 4.5 | 82.1 |
| ● RevCol-T | $224^2$ | 30 | 4.5 | 82.2 |
| ○ Swin-S (Liu et al.) | $224^2$ | 50 | 8.7 | 83.0 |
| ○ MViTv1-B (Fan et al.) | $224^2$ | 37 | 7.8 | 83.0 |
| ○ T2T-ViT-19 (Yuan et al.) | $224^2$ | 39 | 8.4 | 81.4 |
| ● RevBiFPN-S4 (Chiley et al.) | $320^2$ | 49 | 10.6 | 83.0 |
| ● EfficientNet-B5 Tan & Le) | $456^2$ | 30 | 9.9 | 83.6 |
| ● ConvNeXt-S (Liu et al.) | $224^2$ | 50 | 8.7 | 83.1 |
| ● RevCol-S | $224^2$ | 60 | 9.0 | 83.5 |
| ○ Swin-B (Liu et al.) | $224^2$ | 89 | 15.4 | 83.5 |
| ○ DeiT-B (Touvron et al.) | $224^2$ | 86 | 17.5 | 81.8 |
| ○ Rev-ViT-B(Mangalam et al.) | $224^2$ | 87 | 17.6 | 81.8 |
| ● RepLKNet-31B (Ding et al.) | $224^2$ | 79 | 15.3 | 83.5 |
| ● RevBiFPN-S5 (Chiley et al.) | $352^2$ | 82 | 21.8 | 83.7 |
| ● EfficientNet-B6 (Tan & Le) | $528^2$ | 43 | 19.0 | 84.0 |
| ● ConvNeXt-B (Liu et al.) | $224^2$ | 88 | 15.4 | 83.8 |
| ● RevCol-B | $224^2$ | 138 | 16.6 | 84.1 |

| Model | Image Size | Params (M) | FLOPs (G) | Top-1 Acc. |
|---|---|---|---|---|
| *ImageNet-22K pre-trained models (ImageNet-1K fine-tuned)* | | | | |
| ○ Swin-B (Liu et al. | $224^2$ | 88 | 15.4 | 85.2 |
| ○ Swin-B↑ (Liu et al. | $384^2$ | 88 | 47.0 | 86.4 |
| ○ ViT-B↑ (Dosovitskiy et al.) | $384^2$ | 86 | 55.4 | 84.0 |
| ● RepLKNet-31B (Ding et al.) | $224^2$ | 79 | 15.3 | 85.2 |
| ● RepLKNet-31B↑ (Ding et al.) | $384^2$ | 79 | 45.1 | 86.0 |
| ● ConvNeXt-B (Liu et al.) | $224^2$ | 89 | 15.4 | 85.8 |
| ● ConvNeXt-B↑ (Liu et al.) | $384^2$ | 89 | 45.1 | 86.8 |
| ● RevCol-B | $224^2$ | 138 | 16.6 | 85.6 |
| ● RevCol-B↑ | $384^2$ | 138 | 48.9 | 86.6 |
| ○ Swin-L (Liu et al.) | $224^2$ | 197 | 34.5 | 86.3 |
| ○ Swin-L↑ (Liu et al.) | $384^2$ | 197 | 103.9 | 87.3 |
| ○ ViT-L↑ (Dosovitskiy et al.) | $384^2$ | 307 | 190.7 | 85.2 |
| ● RepLKNet-31L (Ding et al.) | $384^2$ | 172 | 96.0 | 86.6 |
| ● ConvNeXt-L (Liu et al.) | $224^2$ | 198 | 34.4 | 86.6 |
| ● ConvNeXt-L↑ (Liu et al.) | $384^2$ | 198 | 101.0 | 87.5 |
| ● RevCol-L | $224^2$ | 273 | 39.0 | 86.6 |
| ● RevCol-L↑ | $384^2$ | 273 | 116.0 | 87.6 |
| ● ConvNeXt-XL↑ (Liu et al.) | $384^2$ | 350 | 179.0 | 87.8 |
| ● RevCol-XL↑ | $384^2$ | 834 | 350.0 | 88.2 |
| *Extra data pre-trained models (ImageNet-1K fine-tuned)* | | | | |
| ● RevCol-XL↑ | $384^2$ | 834 | 350.0 | 89.4 |
| ● RevCol-H↑ | $640^2$ | 2158 | 2537 | 90.0 |

be added to all columns. For all the variants of RevCol, we set the number of compound loss to 3 empirically (eg. for a 8 column RevCol, the intermediate supervision is added to column 2, 4, and 6, and the original classification CE loss is also added to column 8).

The total loss $\boldsymbol{L}$ in is the summation of all intermediate compound loss:

$$\boldsymbol{L} = \sum_{i=1}^{n} (\alpha_i \mathcal{L}_{\text{BCE}} + \beta_i \mathcal{L}_{\text{CE}}) \tag{4}$$

$n$ denotes the total number of compound loss. $\mathcal{L}_{\text{BCE}}$ and $\mathcal{L}_{\text{CE}}$ denotes BCE loss and CE loss correspondingly. $\alpha_i$ and $\beta_i$ are changed linearly with the compound loss number. When the compound loss is added in earlier columns, we use larger value $\alpha_i$ and smaller value $\beta_i$ to keep $I(\boldsymbol{h}, x)$. In later columns, value $\alpha_i$ decreases and $\beta_i$ increases, which helps boost the performance.

## 3 Experiments

We construct different RevCol variants, RevCol-T/S/B/L, to be of similar complexities to Swin transformers and ConvNeXts. We also build a larger RevCol-XL and RevCol-H to test the scaling up capability. These variants adopt different channel dimension $C$, blocks in each column $B$ and column numbers *COL*. The configuration hyper-parameters of these model variants are:

- RevCol-T: $C = (64, 128, 256, 512)$, $B = (2, 2, 4, 2)$, $COL = 4$
- RevCol-S: $C = (64, 128, 256, 512)$, $B = (2, 2, 4, 2)$, $COL = 8$
- RevCol-B: $C = (72, 144, 288, 576)$, $B = (1, 1, 3, 2)$, $COL = 16$
- RevCol-L: $C = (128, 256, 512, 1024)$, $B = (1, 2, 6, 2)$, $COL = 8$
- RevCol-XL: $C = (224, 448, 896, 1792)$, $B = (1, 2, 6, 2)$, $COL = 8$
- RevCol-H: $C = (360, 720, 1440, 2880)$, $B = (1, 2, 6, 2)$, $COL = 8$

We conduct image classification on ImageNet dataset (Deng et al., 2009; Ridnik et al., 2021). We also test our models on the downstream object detection task and semantic segmentation task on

Table 2: **Object detection and semantic segmentation results with different backbones.** *In object detection task*, we report Box AP and mask AP with single scale testing on COCO val. For RevCol-H, we also report multi scale test results (in parentheses). FLOPs are measured under input sizes of (1280, 800). *For semantic segmentation*, we report mIoU results with single/multi-scale testing. FLOPs are measured under input sizes of (2048, 512), (2560, 640) and (2560, 640) for IN-1K, IN-22K and extra data pre-trained models respectively. Our models are highlighted in gray.

| Backbone | Object Detection on COCO | | | | Semantic Segmentation on ADE20K | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | $AP^{box}$ 3× | $AP^{mask}$ 3× | Params (M) | FLOPs (G) | Input Size | $mIoU_{ss}$ (%) | $mIoU_{ms}$ (%) | Params (M) | FLOPs (G) |
| *ImageNet-1K pre-trained* | | | | | | | | | |
| ○ Swin-T (Liu et al.) | 50.5 | 43.7 | 86 | 745 | $512^2$ | 44.5 | 45.8 | 60 | 945 |
| • ConvNeXt-T (Liu et al.) | 50.4 | 43.7 | 86 | 741 | $512^2$ | 46.0 | 46.7 | 60 | 939 |
| • RevCol-T | **50.6** | **43.8** | 88 | 741 | $512^2$ | **47.4** | **47.6** | 60 | 937 |
| ○ Swin-S (Liu et al.) | 51.8 | 44.7 | 107 | 838 | $512^2$ | 47.6 | 49.5 | 81 | 1038 |
| • ConvNeXt-S (Liu et al.) | 51.9 | 45.0 | 108 | 827 | $512^2$ | **48.7** | **49.6** | 82 | 1027 |
| • RevCol-S | **52.6** | **45.5** | 118 | 833 | $512^2$ | 47.9 | 49.0 | 90 | 1031 |
| ○ Swin-B (Liu et al.) | 51.9 | 45.0 | 145 | 982 | $512^2$ | 48.1 | 49.7 | 121 | 1188 |
| • ConvNeXt-B (Liu et al.) | 52.7 | 45.6 | 146 | 964 | $512^2$ | 49.1 | 49.9 | 122 | 1170 |
| • RepLKNet-B (Ding et al.) | 52.2 | 45.2 | 137 | 965 | $512^2$ | **49.9** | **50.6** | 112 | 1170 |
| • RevCol-B | **53.0** | **45.9** | 196 | 988 | $512^2$ | 49.1 | 50.1 | 168 | 1190 |
| *ImageNet-22K pre-trained* | | | | | | | | | |
| ○ Swin-B (Liu et al.) | 53.0 | 45.8 | 145 | 982 | $640^2$ | 50.3 | 51.7 | 121 | 1841 |
| • ConvNeXt-B (Liu et al.) | **54.0** | **46.9** | 146 | 964 | $640^2$ | 52.6 | 53.1 | 122 | 1828 |
| • RepLKNet-B (Ding et al.) | 53.0 | 46.0 | 137 | 965 | $640^2$ | 51.5 | 52.3 | 112 | 1829 |
| • RevCol-B | 53.6 | 46.3 | 196 | 988 | $640^2$ | **52.7** | **53.3** | 168 | 1860 |
| ○ Swin-L (Liu et al.) | 53.9 | 46.7 | 253 | 1382 | $640^2$ | 52.1 | 53.5 | 234 | 2468 |
| • ConvNeXt-L (Liu et al.) | 54.8 | 47.6 | 255 | 1354 | $640^2$ | 53.2 | 53.7 | 235 | 2458 |
| • RepLKNet-L (Ding et al.) | 53.9 | 46.5 | 229 | 1321 | $640^2$ | 52.4 | 52.7 | 207 | 2404 |
| • RevCol-L | **55.6** | **48.2** | 330 | 1453 | $640^2$ | **53.4** | **53.7** | 306 | 2610 |
| *Extra data pre-trained* | | | | | | | | | |
| • RevCol-H | **60.6 (61.0)** | **52.5 (53.0)** | 2407 | 4417 | $640^2$ | **56.9** | **57.1** | 2421 | 20593 |

commonly used COCO and ADE20k dataset. Furthermore, we show the performance of RevCol with transformer on vision and language tasks (shown in Appendix B).

### 3.1 Image Classification

We conduct image classification experiments mainly on ImageNet (Deng et al., 2009) dataset. ImageNet-1K (Deng et al., 2009) contains 1.28M training images and 50K validation images from 1,000 classes. We train RevCol for 300 epochs with 20-epoch warmup using AdamW optimizer (Loshchilov & Hutter, 2017). A total batch size of 4096 with $4e^{-3}$ learning rate and a weight decay of 0.05 are used in training. We adopt linear learning rate scheduler for warmup and cosine decaying scheduler afterward. We include most of the augmentation and regularization strategies of (Liu et al., 2022b) in training. We also pre-train our models on the larger ImageNet-22K dataset (Ridnik et al., 2021), which contains 14.2 million images and 22K classes. More details can be found in Appendix F.

In Tab. 1 Left, we compare our RevCol with commonly used recent Transformers and CNNs on ImageNet-1k. Our RevCol outperforms a large number of Transformers with similar complexities. Compared with ConvNeXt, even we shrink the convolution kernel to $3 \times 3$, our RevCol can still outperform ConvNeXt among all variants. In Tab. 1 Right, we present results with models trained on ImageNet-22k and fine-tuned on ImageNet-1k. The results demonstrate our model has strong ability of scaling large. To further test the scaling up effectiveness of large dataset, we build a 168-million-image semi-labeled dataset (see Appendix D). With extra data pre-training and ImageNet-1k fine-tuning, our RevCol-H achieves **90.0% top-1 accuracy** on ImageNet-1k validation set.

### 3.2 Object Detection

We evaluate the proposed RevCol on object detection task. Experiments are conducted on the MS-COCO (Lin et al., 2014) dataset using the Cascade Mask R-CNN (Cai & Vasconcelos, 2019)

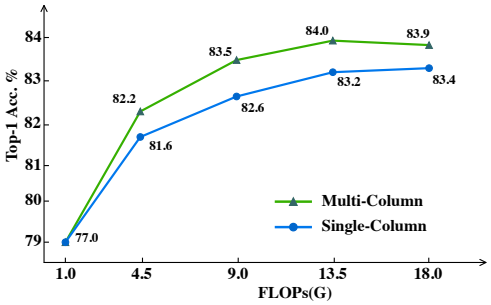


Figure 3: ImageNet-1K performance of maintaining a constant FLOPs of a single column and adding more columns.

Table 3: ImageNet 1K performances of various number of columns in RevCols under the similar computational budget.

| # column | Params | FLOPs | FLOPs per col. | Top-1 Acc. |
|---|---|---|---|---|
| 1 | 28M | 4.4G | 4.40G | 81.9 |
| 4 | 30M | 4.5G | 1.12G | 82.2 |
| 8 | 34M | 4.7G | 0.59G | 82.3 |
| 12 | 33M | 4.4G | 0.35G | 82.2 |
| 20 | 35M | 4.2G | 0.21G | 81.0 |

framework. We load the RevCol models pre-trained on ImageNet-1K and ImageNet-22k for fine-tuning on the downstream object detection task. We adopt the AdamW optimizer and $3\times$ multi-scale training schedule, which is the same as Swin Transformer (Liu et al., 2021). We also adopt layer-wise learning rate decay to alleviate the overfitting of larger models. The intermediate-supervision is not used in fine-tuning on detection tasks. Appendix F gives more training details.

In Tab. 2, we compare the box $\text{AP}_{box}$ and $\text{AP}_{mask}$ with Swin/ConvNeXt in variant sizes on COCO detection validation set. for RevCol-T/S/B/L, we adopt Cascade Mask R-CNN (Cai & Vasconcelos, 2019) as framework and report the single scale test result. For RevCol-H, we adopt HTC++ (Chen et al., 2019) as framework and report results with single scale and multi scale testing. Our multi-column models show significant advantages compared to single-column models. In addition, our largest model RevCol-H achieves **61.2%** $\text{AP}_{box}$ and **53.6%** $\text{AP}_{mask}$ on COCO detection test-dev set, which further validates the favorable transfer ability with a larger model size and larger pre-trained dataset.

### 3.3 Semantic Segmentation

We also evaluate RevCol backbones on the ADE20K semantic segmentation task with UperNet (Xiao et al., 2018) framework. All model variants are trained for 160K iterations with a batch size of 16. We do not use intermediate-supervision in down-stream fine-tune process. Other experimental settings follow BEiT (Bao et al., 2021) and can be found in Appendix F. In Tab. 2, we report validation mIoU with single-scale and multi-scale flip testing. RevCol models can achieve competitive performance across different model capacities, further validating the effectiveness of our architecture design. Pretrained with our extra data, RevCol-H achieves an mIoU of 57.1%, which shows feasible scalability towards large-scale vision applications.

### 3.4 Empirical Analysis

#### 3.4.1 Performance Gain of Reversible Columns Architecture

In this section, we evaluate the performance gain of using reversible columns. In the first experiment, we fix a single column's structure and FLOPs then simply add more columns to scale large and test the performance. At the same time, we plot the vanilla single-column models with according model sizes. As depicted in Fig. 3, compared to single-column models, using multi-column reversible architecture always gets better performance under same FLOPs. Besides, within a certain range, scaling up RevCol in terms of increasing column numbers can have similar performance gains compared to scaling up with both depth and channel numbers(width) in single-column models. In the second experiment, we constrain the total FLOPs to about 4.5G and test models with different column numbers. In other words, we add more columns but scale down the size of a single column. In Tab. 3, we notice that adopt column number at the range of 4 to 12 can maintain the model's performance, while more column models suffer from performance degradation. We believe the reason is the width and depth in a single column are too low to keep representation ability.

#### 3.4.2 Reversible Networks vs. Non-Reversible Networks

In this section, we ablate different design patterns of reversible connections. First, we build a non-reversible multi-column network using the fusion module of HRNet. Second, we build another single

Table 4: Performance comparison on ImageNet-1K of different design patterns. Row-1 represents HRNet style network w/o *reversible connections*. Row-2 represents RevNet style network w/o *multi-column fusions*. Row-3 are our proposed RevCols.

| *rev. conn.* | *multi-col.* | Params | FLOPs | Acc. |
|---|---|---|---|---|
| | ✓ | 35M | 4.9G | 78.8 |
| ✓ | | 34M | 4.5G | 81.6 |
| ✓ | ✓ | 30M | 4.5G | 82.2 |

Table 5: Performance comparison between models with and without *intermediate supervision*. Results are reported on ImageNet-1K and COCO dataset. We use $1\times$ training schedule on COCO detection task.

| Model | *inter. sup.* | Top-1 Acc. | $AP_{box}$ | $AP_{mask}$ |
|---|---|---|---|---|
| RevCol-T | ✗ | 81.4 | 48.3 | 41.8 |
| RevCol-T | ✓ | 82.2 (+0.8) | 48.8 (+0.6) | 42.2 (+0.4) |
| RevCol-S | ✗ | 83.0 | 50.7 | 43.8 |
| RevCol-S | ✓ | 83.5 (+0.5) | 51.1 (+0.4) | 43.8 (+0.0) |
| RevCol-B | ✗ | 83.2 | 51.2 | 44.2 |
| RevCol-B | ✓ | 84.1 (+0.9) | 51.6 (+0.4) | 44.2 (+0.0) |

column reversible ConvNeXt using the design of RevNet as shown in Fig. 2(a). We compare the two designs with our RevCols. The evaluation result is shown in Tab. 4. The non-reversible multi-column network suffers from information loss during propagation, which could result in lower accuracy. The reversible single-column network maintains information during propagation, but lack the superiority of multi-level fusion. This experiment further indicates the effectiveness of combining the reversible design with multi-column networks.

#### 3.4.3 Performance gain of using Intermediate Supervision

In this section, we evaluate the performance of RevCol-T/S/B with and without intermediate supervision on ImageNet-1K. We also evaluate the object detection task performance using $1\times$ training schedule on MS-COCO dataset. Other settings remain the same. From the validation results in Tab. 5, models trained with intermediate supervision achieves 0.5% to 0.9% better top-1 accuracy. Besides, intermediate supervision also benefits down-stream tasks, which further demonstrates its effectiveness.

## 4 Related Works

### 4.1 Disentangle Representation Learning and Part-Whole Hierarchy

A disentangled representation is generally described as one which separates the factors of variation, explicitly representing the important attributes of the data (Desjardins et al., 2012; Bengio et al., 2013). Desjardins et al. (2012); Kulkarni et al. (2015); Higgins et al. (2017); Chen et al. (2016); Karras et al. (2019) seek to learn disentangled representations through generative models. Locatello et al. (2019) points out that unsupervised learning of disentangled representations is fundamentally impossible without inductive biases both on the considered learning approaches and the datasets. The recent proposal of GLOM (Hinton, 2021) gives an idea of representing a part-whole hierarchy by a weight-sharing columns. The GLOM architecture provides an interpretable part-whole hierarchies for deep neural network (Garau et al., 2022). In RevCol, we adopt the design of using columns, but not modeling the process of formulating islands. On the contrary, our column iteration process maintains both low-level and high-level information and gradually disentangle them. Rather than using self-supervised methods, RevCol can be trained with supervision end-to-end.

### 4.2 Reversible Networks

Gomez et al. (2017) firstly propose RevNet that allow back propagation without saving intermediate activations. The reversible design remarkably saves the training cost, since it keep $\mathcal{O}(1)$ GPU memory consumption as model depth scaling up. Jacobsen et al. (2018) propose a fully reversible network that can reverse back to the input without any information loss. Chang et al. (2018) develop a theoretical framework on stability and reversibility of deep neural network and derive reversible networks that can go arbitrarily deep. Mangalam et al. (2022) expand the reversible network scope from CNN to Transformers. RevBiFPN (Chiley et al., 2022), a concurrent work of ours, add the reversible connections to BiFPN (Tan et al., 2020) network. Our RevCol maintains the information without loss inside each column rather than the whole BiFPN network in RevBiFPN.

## REFERENCES


Shaojie Bai, Vladlen Koltun, and J Zico Kolter. Multiscale deep equilibrium models. *Advances in Neural Information Processing Systems*, 33:5238–5250, 2020.

Hangbo Bao, Li Dong, and Furu Wei. Beit: Bert pre-training of image transformers. *arXiv preprint arXiv:2106.08254*, 2021.

Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828, 2013.

Zhaowei Cai and Nuno Vasconcelos. Cascade r-cnn: high quality object detection and instance segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 43(5):1483–1498, 2019.

Rich Caruana. Multitask learning. *Machine learning*, 28(1):41–75, 1997.

Bo Chang, Lili Meng, Eldad Haber, Lars Ruthotto, David Begert, and Elliot Holtham. Reversible architectures for arbitrarily deep residual neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

Kai Chen, Jiangmiao Pang, Jiaqi Wang, Yu Xiong, Xiaoxiao Li, Shuyang Sun, Wansen Feng, Ziwei Liu, Jianping Shi, Wanli Ouyang, et al. Hybrid task cascade for instance segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4974–4983, 2019.

Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. *Advances in neural information processing systems*, 29, 2016.

Vitaliy Chiley, Vithursan Thangarasa, Abhay Gupta, Anshul Samar, Joel Hestness, and Dennis DeCoste. Revbifpn: The fully reversible bidirectional feature pyramid network. *arXiv preprint arXiv:2206.14098*, 2022.

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.

Guillaume Desjardins, Aaron Courville, and Yoshua Bengio. Disentangling factors of variation via generative entangling. *arXiv preprint arXiv:1210.5474*, 2012.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

Xiaohan Ding, Xiangyu Zhang, Jungong Han, and Guiguang Ding. Scaling up your kernels to 31x31: Revisiting large kernel design in cnns. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11963–11975, 2022a.

Xiaohan Ding, Xiangyu Zhang, Jungong Han, and Guiguang Ding. Scaling up your kernels to 31x31: Revisiting large kernel design in cnns. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11963–11975, 2022b.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

Haoqi Fan, Bo Xiong, Karttikeya Mangalam, Yanghao Li, Zhicheng Yan, Jitendra Malik, and Christoph Feichtenhofer. Multiscale vision transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 6824–6835, 2021.

Nicola Garau, Niccolò Bisagno, Zeno Sambugaro, and Nicola Conci. Interpretable part-whole hierarchies and conceptual-semantic relationships in neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 13689–13698, 2022.

Golnaz Ghiasi, Barret Zoph, Ekin D Cubuk, Quoc V Le, and Tsung-Yi Lin. Multi-task self-training for learning general representations. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 8856–8865, 2021.

Aidan N Gomez, Mengye Ren, Raquel Urtasun, and Roger B Grosse. The reversible residual network: Backpropagation without storing activations. *Advances in neural information processing systems*, 30, 2017.

Qi Han, Zejia Fan, Qi Dai, Lei Sun, Ming-Ming Cheng, Jiaying Liu, and Jingdong Wang. On the connection between local attention and dynamic depth-wise convolution. In *International Conference on Learning Representations*, 2021.

Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 16000–16009, 2022.

Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. beta-VAE: Learning basic visual concepts with a constrained variational framework. In *International Conference on Learning Representations*, 2017. URL `https://openreview.net/forum?id=Sy2fzU9gl`.

Geoffrey Hinton. How to represent part-whole hierarchies in a neural network. *arXiv preprint arXiv:2102.12627*, 2021.

Jörn-Henrik Jacobsen, Arnold Smeulders, and Edouard Oyallon. i-revnet: Deep invertible networks. *arXiv preprint arXiv:1802.07088*, 2018.

Peng-Tao Jiang, Chang-Bin Zhang, Qibin Hou, Ming-Ming Cheng, and Yunchao Wei. Layercam: Exploring hierarchical class activation maps for localization. *IEEE Transactions on Image Processing*, 30:5875–5888, 2021.

Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 4401–4410, 2019.

Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Joan Puigcerver, Jessica Yung, Sylvain Gelly, and Neil Houlsby. Big transfer (bit): General visual representation learning. In *European conference on computer vision*, pp. 491–507. Springer, 2020.

Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey Hinton. Similarity of neural network representations revisited. In *International Conference on Machine Learning*, pp. 3519–3529. PMLR, 2019.

Tejas D Kulkarni, William F Whitney, Pushmeet Kohli, and Josh Tenenbaum. Deep convolutional inverse graphics network. *Advances in neural information processing systems*, 28, 2015.

Chen-Yu Lee, Saining Xie, Patrick Gallagher, Zhengyou Zhang, and Zhuowen Tu. Deeply-supervised nets. In *Artificial intelligence and statistics*, pp. 562–570. PMLR, 2015.

Timothy P Lillicrap, Adam Santoro, Luke Marris, Colin J Akerman, and Geoffrey Hinton. Backpropagation and the brain. *Nature Reviews Neuroscience*, 21(6):335–346, 2020.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pp. 740–755. Springer, 2014.

Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2117–2125, 2017.

Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 10012–10022, 2021.

Ze Liu, Han Hu, Yutong Lin, Zhuliang Yao, Zhenda Xie, Yixuan Wei, Jia Ning, Yue Cao, Zheng Zhang, Li Dong, et al. Swin transformer v2: Scaling up capacity and resolution. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 12009–12019, 2022a.

Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A convnet for the 2020s. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11976–11986, 2022b.

Francesco Locatello, Stefan Bauer, Mario Lucic, Gunnar Raetsch, Sylvain Gelly, Bernhard Schölkopf, and Olivier Bachem. Challenging common assumptions in the unsupervised learning of disentangled representations. In *international conference on machine learning*, pp. 4114–4124. PMLR, 2019.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

Dhruv Mahajan, Ross Girshick, Vignesh Ramanathan, Kaiming He, Manohar Paluri, Yixuan Li, Ashwin Bharambe, and Laurens Van Der Maaten. Exploring the limits of weakly supervised pretraining. In *Proceedings of the European conference on computer vision (ECCV)*, pp. 181–196, 2018.

Karttikeya Mangalam, Haoqi Fan, Yanghao Li, Chao-Yuan Wu, Bo Xiong, Christoph Feichtenhofer, and Jitendra Malik. Reversible vision transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10830–10840, 2022.

Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

Myle Ott, Sergey Edunov, David Grangier, and Michael Auli. Scaling neural machine translation. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pp. 1–9, Brussels, Belgium, October 2018. Association for Computational Linguistics. doi: 10.18653/v1/W18-6301.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning*, pp. 8748–8763. PMLR, 2021.

Tal Ridnik, Emanuel Ben-Baruch, Asaf Noy, and Lihi Zelnik-Manor. Imagenet-21k pretraining for the masses. *arXiv preprint arXiv:2104.10972*, 2021.

Sebastian Ruder. An overview of multi-task learning in deep neural networks. *arXiv preprint arXiv:1706.05098*, 2017.

Ozan Sener and Vladlen Koltun. Multi-task learning as multi-objective optimization. *Advances in neural information processing systems*, 31, 2018.

Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. In *54th Annual Meeting of the Association for Computational Linguistics*, pp. 1715–1725. Association for Computational Linguistics (ACL), 2016.

Shuai Shao, Zeming Li, Tianyuan Zhang, Chao Peng, Gang Yu, Xiangyu Zhang, Jing Li, and Jian Sun. Objects365: A large-scale, high-quality dataset for object detection. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 8430–8439, 2019.

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1–9, 2015.

Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International conference on machine learning*, pp. 6105–6114. PMLR, 2019.

Mingxing Tan, Ruoming Pang, and Quoc V Le. Efficientdet: Scalable and efficient object detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 10781–10790, 2020.

Bart Thomee, David A Shamma, Gerald Friedland, Benjamin Elizalde, Karl Ni, Douglas Poland, Damian Borth, and Li-Jia Li. Yfcc100m: The new data in multimedia research. *Communications of the ACM*, 59(2):64–73, 2016.

Naftali Tishby and Noga Zaslavsky. Deep learning and the information bottleneck principle. In *2015 ieee information theory workshop (itw)*, pp. 1–5. IEEE, 2015.

Naftali Tishby, Fernando C Pereira, and William Bialek. The information bottleneck method. *arXiv preprint physics/0004057*, 2000.

Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. *arXiv preprint arXiv:2012.12877*, 2020.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Jingdong Wang, Ke Sun, Tianheng Cheng, Borui Jiang, Chaorui Deng, Yang Zhao, Dong Liu, Yadong Mu, Mingkui Tan, Xinggang Wang, et al. Deep high-resolution representation learning for visual recognition. *IEEE transactions on pattern analysis and machine intelligence*, 43(10):3349–3364, 2020.

Wenhui Wang, Hangbo Bao, Li Dong, Johan Bjorck, Zhiliang Peng, Qiang Liu, Kriti Aggarwal, Owais Khan Mohammed, Saksham Singhal, Subhojit Som, et al. Image as a foreign language: Beit pretraining for all vision and vision-language tasks. *arXiv preprint arXiv:2208.10442*, 2022.

Yulin Wang, Zanlin Ni, Shiji Song, Le Yang, and Gao Huang. Revisiting locally supervised learning: an alternative to end-to-end training. *arXiv preprint arXiv:2101.10832*, 2021.

Tete Xiao, Yingcheng Liu, Bolei Zhou, Yuning Jiang, and Jian Sun. Unified perceptual parsing for scene understanding. In *Proceedings of the European conference on computer vision (ECCV)*, pp. 418–434, 2018.

Zhenda Xie, Zheng Zhang, Yue Cao, Yutong Lin, Jianmin Bao, Zhuliang Yao, Qi Dai, and Han Hu. Simmim: A simple framework for masked image modeling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9653–9663, 2022.

I Zeki Yalniz, Hervé Jégou, Kan Chen, Manohar Paluri, and Dhruv Mahajan. Billion-scale semi-supervised learning for image classification. *arXiv preprint arXiv:1905.00546*, 2019.

Li Yuan, Yunpeng Chen, Tao Wang, Weihao Yu, Yujun Shi, Zi-Hang Jiang, Francis EH Tay, Jiashi Feng, and Shuicheng Yan. Tokens-to-token vit: Training vision transformers from scratch on imagenet. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 558–567, 2021a.

Lu Yuan, Dongdong Chen, Yi-Ling Chen, Noel Codella, Xiyang Dai, Jianfeng Gao, Houdong Hu, Xuedong Huang, Boxin Li, Chunyuan Li, et al. Florence: A new foundation model for computer vision. *arXiv preprint arXiv:2111.11432*, 2021b.

Amir R Zamir, Alexander Sax, William Shen, Leonidas J Guibas, Jitendra Malik, and Silvio Savarese. Taskonomy: Disentangling task transfer learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3712–3722, 2018.

Xiaohua Zhai, Alexander Kolesnikov, Neil Houlsby, and Lucas Beyer. Scaling vision transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 12104–12113, 2022.

Yuanhan Zhang, Qinghong Sun, Yichun Zhou, Zexin He, Zhenfei Yin, Kun Wang, Lu Sheng, Yu Qiao, Jing Shao, and Ziwei Liu. Bamboo: Building mega-scale vision dataset continually with human-machine synergy, 2022.

Bolei Zhou, Agata Lapedriza, Aditya Khosla, Aude Oliva, and Antonio Torralba. Places: A 10 million image database for scene recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017.

# A MICRO DESIGN DETAILS

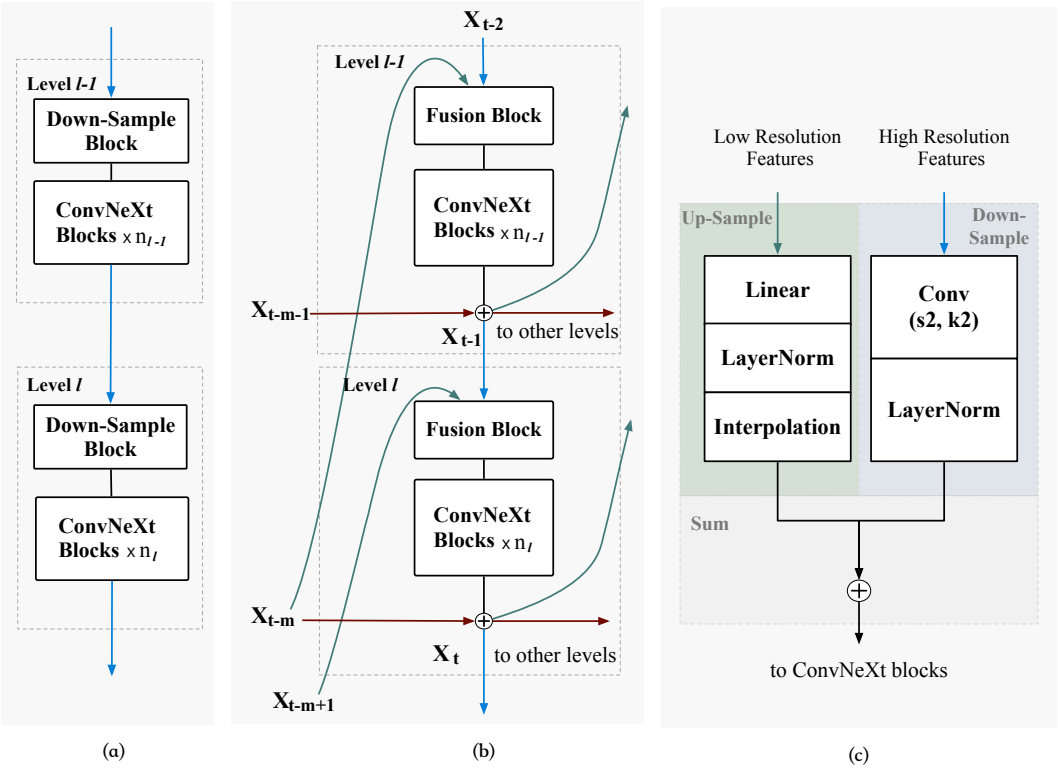


Figure 4: (a) Levels in ConvNeXt. Level $l$ contains a patch merging down-sample block and $n_l$ residual blocks. (b) Levels in RevCol. Level $l$ is composed of a fusion module, $n_l$ residual blocks and a reversible operation. Note that Level $l$ takes features maps $x_{t-1}$, $x_{t-m+1}$ and $x_{t-m}$ as input. Feature maps $x_{t-1}$ and $x_{t-m+1}$ are fed into the fusion module and feature maps $x_{t-m}$ are fed into the reversible operation. (c) Design of the fusion module.

In this section, we provide the architecture design details for RevCol. As depicted in Fig. 2 and Section 2.2, our RevCol contains multiple columns with reversible connections. Fig. 4 (a) shows the architecture of ConvNeXt. Note that we replace the $7 \times 7$ depth-wise convolution in ConvNeXt with $3 \times 3$, as described in Sec. 2.2.2. In Fig. 4 (b), we show in detail how to extend to our RevCol on the basis of ConvNeXt. First, we replace the down-sample block with a fusion block to fuse low-level representations in current column and high-level ones from the previous column, and Fig. 4 (c) shows the details of fusion block which contains up-sample and down-sample operations to handle different resolutions. Second, for each level, same-level representations from the previous column are added to current level's output and are ready to propagate as a whole. Thanks to the two modifications, feature maps from different hierarchies aggregate together to form the intermediate representation. In Fig. 4 (c), we use a `Linear-LayerNorm` followed by a nearest interpolation to up-sample low resolution features. A $2 \times 2$ kernel `Conv2d` with stride 2 down-samples the high resolution features, followed by a `LayerNorm` to balance the contributions of the two inputs.

# B GENERALIZATION TO TRANSFORMERS

## B.1 VISION TRANSFORMER MODELS

RevCol contains multiple light-weight sub-networks with reversible connections. In this paper, we adopt the ConvNext micro design by default except for multi-columns fusion and smaller convolution kernel as described in Sec. 2.2.2. However, the micro design of our RevCol is not limited to convolutional networks, but is also compatible with isotropic designing, such as the vanilla vision

transformer (ViT) (Dosovitskiy et al., 2020). In this section, we show the micro design of RevCol can generalized to vanilla ViT, namely RevCol-ViT, with promising experimental results.

net-ViT maintains the feature resolution in the reversible columns. Thus the patch merging blocks and up-sample blocks in the fusion modules are replaced with a simple linear projection with a post LayerNorm. We use the vanilla ViT building block instead of the ConvNext building block variant. The post LayerNorms and normalized dot-product attention are used in ViT blocks to stabilize training convergence, similar to Liu et al. (2022a). With the properties of isotropy, we evenly arrange the building blocks in each column. The configuration details of RevCol-ViT are:

- RevCol-ViT-S: $C = (224, 224, 224, 224)$, $B = (2, 2, 2, 2)$, $\mathit{HEAD} = 4$, $\mathit{COL} = 4$
- RevCol-ViT-B: $C = (384, 384, 384, 384)$, $B = (3, 3, 3, 3)$, $\mathit{HEAD} = 6$, $\mathit{COL} = 4$

Table 6: ImageNet-1K classification results. We compare our RevCol-ViT with state-of-the-art isotropic ○ Vision Transformers and ● CNNs that have comparable FLOPs and parameters.

| Model | Image Size | Params | FLOPs | Top-1 Acc. |
|---|---|---|---|---|
| ○ DeiT-S (Touvron et al., 2020) | $224^2$ | 22M | 4.6G | 79.8 |
| ● ConvNext-S (*iso.*) (Liu et al., 2022b) | $224^2$ | 22M | 4.3G | 79.7 |
| ○ **RevCol-ViT-S** | $224^2$ | 16M | 4.6G | **80.6** |
| ○ ViT-B (Dosovitskiy et al., 2020) | $384^2$ | 86M | 55.4G | 77.9 |
| ○ DeiT-B (Touvron et al., 2020) | $224^2$ | 86M | 17.6G | 81.7 |
| ○ Rev-ViT-B (Mangalam et al., 2022) | $224^2$ | 87M | 17.6G | 81.8 |
| ○ Rev-MViT-B (Mangalam et al., 2022) | $224^2$ | 39M | 8.7G | 82.5 |
| ● ConvNext-B (*iso.*) (Liu et al., 2022b) | $224^2$ | 87M | 16.9G | 82.0 |
| ○ **RevCol-ViT-B** | $224^2$ | 67M | 18.8G | **82.7** |

We use the same training setting with the anisotropic RevCol as described in Sec. 3.1, except that the intermediate supervision is discarded for simplicity and the stochastic depth rate is set as 0.2 for RevCol-B. We scale down the value of last linear projection layers in each FFN accroding to the network depth in initialization, same as BEiT (Bao et al., 2021). In Tab. 6, we compare the RevCol-ViT with vanilla ViT and other concurrent isotropic designs. Our RevCol-ViT surpasses vanilla vision transformer (77.9% for ViT and 81.7% for DeiT) and convolutional network ConvNeXt (82.0%) that have comparable model parameters and computational overhead on ImageNet-1k classification w.r.t. the top-1 accuracy.

### B.2 Language Models

Considering the great success of applying transformer to computer vision, i.e., ViT (Dosovitskiy et al., 2020), we also made some exploration to generalize RevCol to natural language processing (NLP). Based on the design in Appendix B.1, we can easily apply the isotropic RevCol to language models with minor modification. To be specific, we replace the stem module in our RevCol with word embedding and positional encoding in transformer. Then, the RevCol can be plugged into the original transformer as an encoder. The output of the last column in RevCol will be used as the memory keys and values for the attention layers in decoder, just exactly the same as the original transformer.

We select the translation task to evaluate the potential of the RevCol in NLP. We run experiments on the WMT'16 English-German (En-De) dataset with 4.5M sentences and larger WMT'14 English-French dataset with 36M sentences. Each sentence is encoded by joint source and target byte pair encoding following Sennrich et al. (2016). The details of model architecture and the BLEU score are shown in Tab. 7.

All the dataset preparation and the training configurations follows Ott et al. (2018) and the open source project `fairseq`. The models were trained for 300K steps with batch-size of 28,672 tokens on En-De and 200K steps with batch-size of 86,016 on En-Fr. We discard the intermediate supervision for simplicity. As shown in Tab. 7, our RevCol outperforms vanilla transformer with comparable parameters on En-De (28.67 *vs*. 28.43) and En-Fr (43.40 *vs*. 43.07), which demonstrates the RevCol's applicability to NLP.

Table 7: BLEU score on newstest2014 for WMT English-German (En-De) and English-French (En-Fr) translation task. † indicates we re-run the experiments with `fairseq`.

| Model | Encoder | | | | Decoder | | | | Params | Task | BLEU |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | arch | $d_{model}$ | $d_{ff}$ | head | arch | $d_{model}$ | $d_{ff}$ | head | | | |
| Transformer$^{\dagger}_{big}$ (Vaswani et al., 2017) | $N = 6$ | 1024 | 4096 | 16 | $N = 6$ | 1024 | 4096 | 16 | 209M<br>221M | En-De<br>En-Fr | 28.43<br>43.07 |
| **RevCol-Transformer** | $B = (1,1,1,1)$<br>$COL = 4$ | 768 | 3072 | 12 | $N = 6$ | 768 | 3072 | 12 | 200M<br>209M | En-De<br>En-Fr | **28.67**<br>**43.40** |

# C More Analysis Experiments

## C.1 Ablation of kernel size in convolutions

In original ConvNeXt, large kernel convolution achieves in better performance. We conduct same experiments in RevCol-T. As shown in Tab. 8, for 4 column tiny models, using $5 \times 5$ convolution increase the ImageNet-1k Top-1 accuracy by 0.3% and the COCO AP by 0.6 for RevCol-T model. Larger kernel convolution has no more accuracy gain. We consider the multi-level fusion structure already generate large receptive field and this limit the accuracy gain of using large kernel convolution. Therefore, we fix the convolution kernel size to 3 in all variants of our model.

## C.2 GPU Memory Consumption vs Model size

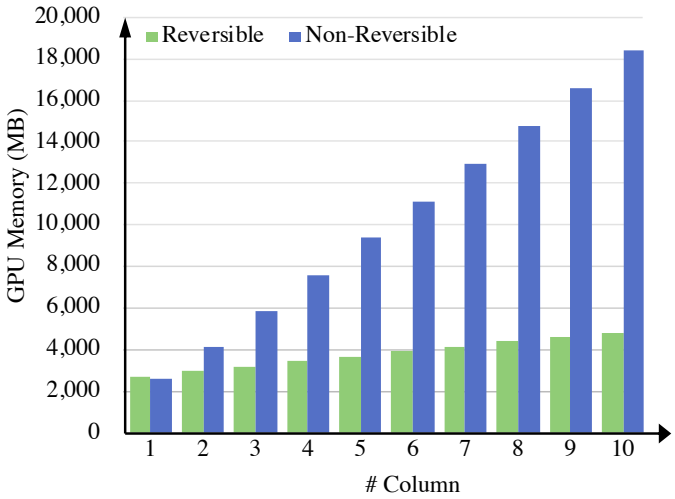


Figure 5: GPU Memory Consumption *vs*. Model size

Fig. 5 plots the GPU memory consumption with the scaling of model size. We fix the computation complexity of a single column to 1G FLOPs and only increase column number. Meanwhile, we

Table 8: Performance of models with larger kernel size convolution.

| Kernel Size | FLOPs | Top-1 Acc | $AP_{box}$ 1× | $AP_{mask}$ 1× |
|---|---|---|---|---|
| 3 | 4.5G | 82.2 | 48.9 | 42.2 |
| 5 | 4.5G | 82.2 | 49.5 | 42.6 |
| 7 | 4.6G | 82.3 | 49.3 | 42.4 |
| 11 | 4.6G | 82.5 | 49.5 | 42.6 |

measure the memory consumption in training process which includes the forward and backward propagation. Our experiments are conducted on Nvidia Tesla V100 GPU under batch-size 64, fp16 precision and PyTorch implementation. With the increase of model size, we can see our RevCol keeps an $\mathcal{O}(1)$ GPU memory consumption, while non-reversible architecture's memory consumption increase linearly with column number. Note that our RevCol does not keep strictly the same GPU memory consumption as column number increase, as reversible networks need to save the operation weights in need for calculating gradients and the re-construction of feature maps in backward propagation.

### C.3 Robustness of the number of columns

In the ablation analysis of the paper, we show that when fix the total FLOPs and add more columns of RevCol, the performance first increases and then get saturated. When the number of columns is extreme large, such as 20, the performance drop because of the representation ability of single column is limited. When the number of columns is usual, such as 4$\tilde{1}$2, the performances are similar, which verifies the setting robustness of the number of columns.

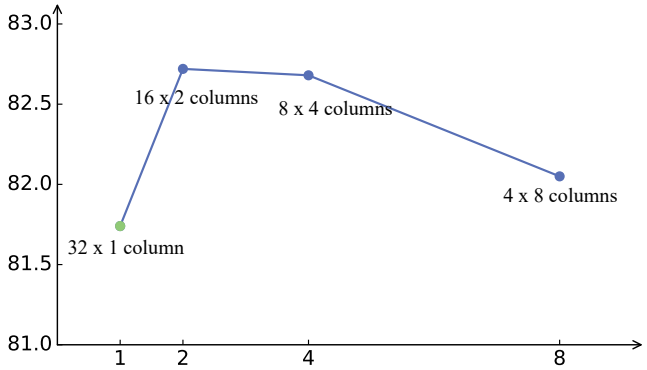


Figure 6: ImageNet top-1 accuracy of different variants of RevCol-ViT-B. Each variant has the same total number of residual blocks and channel dimension.

To further analyze the robustness of the number of columns, in this section, we build some RevCol-ViT-B variants (see Appendix B for more details). Each variant has the same number of residual blocks with the same channel dimension, but different number of columns. In other worlds, these variants have the same channel dimension and different depth of each columns and different number of columns. We use 32 residual blocks totally and maintain the FLOPs about 18G. Fig. 6 show the performance on ImageNet-1K of different variants. The number of columns are 1, 2, 4, and 8, accordingly the depth of each column are 32, 16, 8, and 4. The performance of single column variant is lower (similar to DeiT-B (Touvron et al., 2020)) because of the single column ViT can not maintain the information as multi reversible columns. The performance is decreasing when the number of columns became larger, because of the depth of each columns is not enough. This phenomenon indicates us that given a target FLOPs, the setting of the number of columns is robust unless the depth of each columns or channel dimension is too small.

## D Semi-labeled Privately Collected Dataset for Large Models

### D.1 Data Collection and Pseudo Label System

The dataset consists of around 168 million(M) images, 50M of which labeled and the remaining 118M unlabeled. The majority of labeled images come from public datasets, *e.g.* ImageNet, Places365 (Zhou et al., 2017), and Bamboo (Zhang et al., 2022). The others are web-crawled images annotated by in-door employees. Unlabeled images come from weakly-annotated image-text datasets like YFCC-100M (Thomee et al., 2016). We do not use text annotations.

In order to utilize images of different label domains and the massive unlabeled images, we employ a multi-target label system similar to Ding et al. (2022a) and Ghiasi et al. (2021). We adopt a semi-

supervised learning strategy with ViTs, thus generating *pseudo* labels with continuously increased quality. We only store soft predictions with confidence higher than 1% to save storage. The final version of *pseudo* label we use are generated by a multi-head ViT-Huge teacher, which has an 89.0% ImageNet-1k accuracy.

### D.2 IMAGE DEDUPLICATION

Since the dataset contains large amount of unverified web-crawled images, there are probably validation or test images sneaking into our training dataset. Works like Mahajan et al. (2018) and Yalniz et al. (2019) all regard image deduplication an important procedure for fair experiments.

We first iterate over the entire dataset to filter out suspicious duplicates together with the corresponding test images based on their *pseudo* label distance. This brings more than 10,000 images with high suspicion. We look at these image pairs and finally find about 1,200 exact-duplicates and near-duplicates. Fig. 7 shows some examples of the near-duplicates, which are difficult to detect. Never the less, training a model without removing these duplicates gives less than 0.1% accuracy gain on ImageNet-1k in our experiments. We attribute this to the absence of true labels from these duplicates.

Figure 7: Top: Near duplicates found in unlabeled images. Bottom: ImageNet-1k validation images.

## E SYSTEM-LEVEL COMPARISON WITH SOTA FOUNDATION MODELS

Foundation models (Kolesnikov et al., 2020; Radford et al., 2021; Yuan et al., 2021b) are general-purpose backbones pre-trained on massive and divergent data source. They can adapt to various down-stream tasks with limited domain-specific data in few-shot or even zero-shot manner. Though our RevCol is pure convolutional and pre-trained on single modality dataset, we are competitive among various public SOTAs including Vision Transformers and Vision-Language models, namely, ViT-G (Zhai et al., 2022), SwinV2 (Liu et al., 2022a), BEiT3 (Wang et al., 2022), and Florence (Yuan et al., 2021b), shown in Tab. 9.

Table 9: System-level comparison of state-of-the-art visual foundation models with large-scale pretraining. We include ○ Vision Transformers, ● CNNs, and ● hybrid architectures pretrained either *unsupervised* or *supervised* on *image-only* and *vision-language* datasets. COCO scores marked with † means intermediate fine-tuned on extra data like Object365 (Shao et al., 2019).

| Model | Params | Dataset & Pretrain | | | ImageNet | | COCO test-dev | | ADE20K | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | Images | Annotation | Method | 1K | v2 | $AP_{box}$ | $AP_{mask}$ | mIoU | +ms |
| ○ ViT-G/14 | 1.8 G | 3,000 M | labeled | sup. | 90.5 | 83.3 | - | - | - | - |
| ○ SwinV2-G | 3.0 G | 70 M | labeled | unsup. & sup. | 90.2 | 84.0 | 63.1† | 54.4† | 59.3 | 59.9 |
| ○ BEiT3 | 1.0 G | 35 M | labeled & image-text | unsup. & sup. | 89.6 | - | 63.7† | 54.8† | 62.0 | 62.8 |
| ● Florence | 0.9 G | 900 M | image-text | sup. | 90.1 | - | 62.4 | - | - | - |
| ● RevCol-H | 2.1 G | 168 M | semi-labeled | sup. | 90.0 | 84.0 | 61.2 | 53.6 | 56.8 | 57.1 |

# F MORE TRAINING DETAILS

This section gives more training details on ImageNet classification, COCO detection, and ADE20K segmentation.

## F.1 INTERMEDIATE SUPERVISION SETTINGS

We add intermediate supervision in ImageNet-1k training, ImageNet-22k and extra data pre-training. We used a 4-blocks decoder with gradually up-sampled feature maps in ImageNet-1k training. The block setting remains the same as Sec. 2.2 We use a single layer decoder in ImageNet-22k and extra data pre-training. For all the variants of RevCol, we set the number of compound loss $n$ to 3 empirically (eg. for a 8 column RevCol, the intermediate supervision is added to column 2, 4, and 6, and the original classification CE loss is also added to column 8). $\alpha_i$ is set to 3,2,1 and $\beta_i$ is set to 0.25, 0.5, 0.75.

## F.2 HYPERPARAMETERS USED FOR TRAINING AND PRE-TRAINING

This section introduces the training details for main experiments, the supervised training on ImageNet and extra data. We show this setting in Tab. 10. All experiments in ablation studies are superivised trained on ImageNet-1K except additional descriptions and also follow settings described in this section.

| **Hyperparameters** | **ImageNet-1K** | **ImageNet-22K** | **168M Extra Data** |
|---|---|---|---|
| | **T/S/B** | **B/L/XL** | **XL/H** |
| Input resolution | $224^2$ | | $224^2$ |
| Training epochs | 300 | 90 | 10 |
| Warmup epochs | 20 | 5 | 0.15 |
| Batch size | 4096 | | 5120 |
| Peak learning rate | 4e-3 | 5e-4 | 6.25e-4 |
| Learning rate schedule | cosine | | cosine |
| Layer-wise learning rate decay | ✗ | | ✗ |
| AdamW momentum | (0.9, 0.999) | | (0.9, 0.999) |
| Weight decay | 0.05 | 0.1 | 0.05 |
| Gradient clipping | ✗ | | 1.0 (element-wise) |
| Drop path | 0.1/0.3/0.4 | 0.3 | 0.2 |
| EMA | 0.9999 | ✗ | ✗ |
| Label smoothing $\varepsilon$ | 0.1 | | 0.1 |
| Data augment | RandAug (9, 0.5) | | RandAug (9, 0.5) |
| Mixup | 0.8 | | ✗ |
| CutMix | 1.0 | | ✗ |
| Random erase | 0.25 | | ✗ |

Table 10: Hyperparameters for training and pre-training RevCol.

## F.3 HYPERPARAMETERS USED FOR FINE-TUNING

This section gives the hyperparameters used for fine-tuning on ImageNet-1K and downstrea COCO object detection and instance segmentation, ADE20K semantic segmentation tasks, as shown in Tab. 11, Tab. 12 and Tab. 12.

| Hyperparameters | ImageNet-1K |
|---|---|
| | B/L/XL/H |
| Input resolution | $384^2$/$384^2$/$384^2$/$640^2$ |
| Fine-tuning epochs | 30 |
| Warmup epochs | 0 |
| Batch size | 512 |
| Peak learning rate | 5e-5 |
| Layer-wise learning rate decay | 0.9/0.8/0.8/0.8 |
| AdamW momentum | (0.9, 0.999) |
| Weight decay | 1e-8 |
| Learning rate schedule | cosine |
| Head init scale | 0.001 |
| Drop path | 0.2/0.3/0.4/0.5 |
| EMA | ✗/✗/✗/0.9999 |
| Gradient clipping | 10.0 (norm) |
| Label smoothing $\varepsilon$ | 0.1 |
| Data augment | RandAug (9, 0.5) |
| Mixup | ✗ |
| CutMix | ✗ |
| Random erase | 0.25 |

Table 11: Hyperparameters for fine-tuning RevCol on ImageNet-1K classification.

| Hyperparameters | IN-1K Pre-trained | IN-22K Pre-trained |
|---|---|---|
| | RevCol-T/S/B | RevCol-B/L |
| Input resolution | $512^2$ | $640^2$ |
| Fine-tuning steps | 80k | |
| Batch size | 16 | |
| Peak learning rate | 4e-5 | |
| Warmup steps | 1500 | |
| Layer-wise learning rate decay | 1.0 | 0.9 |
| AdamW momentum | (0.9, 0.999) | |
| Weight decay | 0.01 | |
| Drop path | 0.3 | |

Table 12: Hyperparameters for fine-tuning RevCol on semantic segmentation.

| Hyperparameters | IN-1K Pre-trained | IN-22K Pre-trained |
|---|---|---|
| | RevCol-T/S/B | RevCol-B/L |
| Fine-tuning epochs | 36 | |
| Batch size | 16 | |
| Peak learning rate | 2e-4 | 1e-4 |
| Warmup steps | 1500 | |
| Layer-wise learning rate decay | 0.85/0.8/0.8 | 0.8/0.7 |
| AdamW momentum | (0.9, 0.999) | |
| Weight decay | 0.05 | |
| Drop path | 0.3/0.4/0.4 | 0.6/0.7 |

Table 13: Hyperparameters for fine-tuning RevCol on object detection.

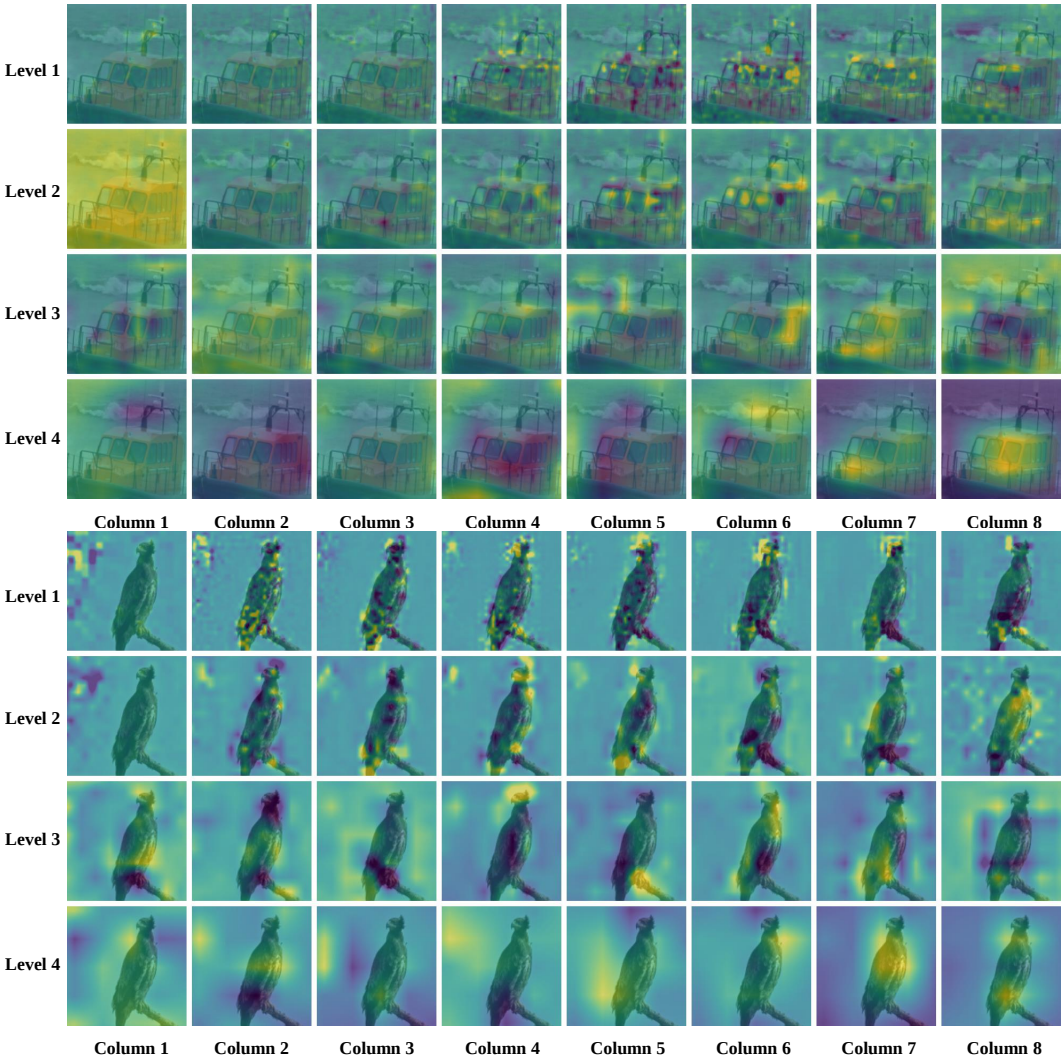


Figure 8: Visualizations of class activation maps using LayerCAM (Jiang et al., 2021) for different levels and columns.

## G VISUALIZATIONS OF FEATURE DISENTANGLEMENT

In this section, we show our RevCol can disentangle features with stacked columns, which is different from the conventional sequential networks. We use RevCol-S pre-trained on ImageNet-1K for analysis. First, we visualize the class activation maps (CAMs) for outputs of each last layer of a level. We adopt LayerCAM (Jiang et al., 2021) technology to generate the CAMs with the predicted classes. Fig. 8 show the heatmaps of activation. With the levels and columns going deeper, the features focus on the regions with more semantics. The outputs of RevCol-S are the different levels of last column. These features with high level semantics focus on different parts of the image and the whole part of the object, achieving disentanglement of features for task-relevant and task-irrelevant.

To quantify the disentanglement, we use Centered Kernel Alignment (CKA) similarity metric (Kornblith et al., 2019) to measure the similarity between representations in RevCol-S. We calculate the CKA similarities between intermediate features in different levels and columns and images or labels of each category in the ImageNet val set. Then we plot the similarities of the category with the highest label similarity in Fig. 9. As shown in the figure, the similarities between images and intermediate features are not clearly distinguished at different levels in Column 2-5, while the features

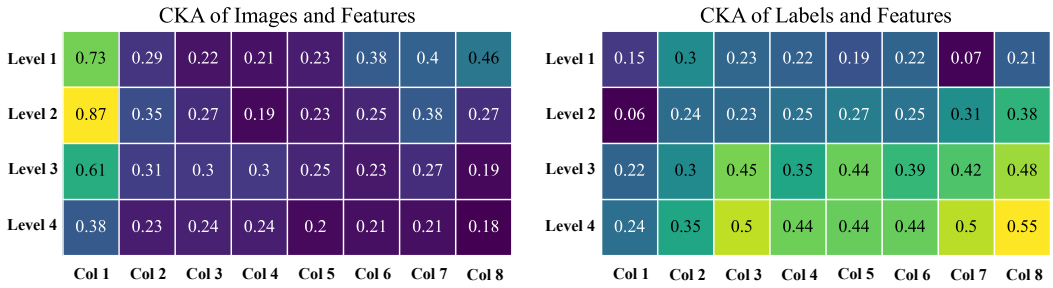


Figure 9: CKA similarities (Kornblith et al., 2019) of features and images/labels for different levels and columns.

with higher levels have lower similarity to the images in Column 6-8. The similarities between labels and intermediate features are also more distinct in higher columns.