

# Auto-Compressing Networks

**Vaggelis Dorovatas**

*School of Electrical and Computer Engineering,  
National Technical University of Athens*

*el19171@mail.ntua.gr*

**Georgios Paraskevopoulos**

*School of Electrical and Computer Engineering,  
National Technical University of Athens*

*geopar@central.ntua.gr*

**Alexandros Potamianos**

*School of Electrical and Computer Engineering,  
National Technical University of Athens*

*apotam@central.ntua.gr*

## Abstract

Deep neural networks with short residual connections have demonstrated remarkable success across domains, but increasing depth often introduces computational redundancy without corresponding improvements in representation quality. In this work, we introduce Auto-Compressing Networks (ACNs), an architectural variant where additive long feedforward connections from each layer to the output replace traditional short residual connections. By analyzing the distinct dynamics induced by this modification, we reveal a unique property we coin as *auto-compression*—the ability of a network to organically compress information during training with gradient descent, through architectural design alone. Through auto-compression, information is dynamically "pushed" into early layers during training, enhancing their representational quality and revealing potential redundancy in deeper ones, resulting in a sparse yet powerful network at inference. We theoretically show that this property emerges from layer-wise training patterns present in ACNs, where layers are dynamically utilized during training based on task requirements. We also find that ACNs exhibit enhanced noise robustness compared to residual networks, superior performance in low-data settings, improved transfer learning capabilities, and mitigate catastrophic forgetting suggesting that they learn representations that generalize better despite using fewer parameters. Our results demonstrate up to 18% reduction in catastrophic forgetting and 30-80% architectural compression while maintaining accuracy across vision transformers, MLP-mixers, and BERT architectures. Furthermore, we demonstrate that when coupling ACNs with traditional pruning techniques, the compression gain persists and enables significantly better sparsity-performance trade-offs compared to conventional architectures. These findings establish ACNs as a practical approach to developing efficient neural architectures that automatically adapt their computational footprint to task complexity, while learning robust representations suitable for noisy real-world tasks and continual learning scenarios.

## 1 Introduction

Deep learning has achieved significant breakthroughs across diverse tasks and domains (Krizhevsky et al., 2012; LeCun et al., 2015; Brown et al., 2020); however, it still lacks the flexibility, robustness, and efficiency of biological networks. Modern models rely on deep architectures with billions of parameters, leading to high computational, storage, and energy costs. Architecturally, these large models are primarily characterized by short residual connections (He et al., 2016), a design initially developed to enable robust training of deep neural networks via backpropagation. These skip connections establish a network topology where multiple information pathways are created (Veit et al., 2016), resulting in an ensemble-like behavior that delivers more efficient training and superior generalization compared to traditional feedforward networks.

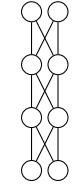
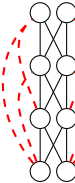

Arch	Connectivity	Forward Propagation	Backward (Gradient) Propagation
FFN		$y_F = \prod_{i=1}^L w_i x_0$	$\frac{\partial y_F}{\partial w_i} = \underbrace{\left( \prod_{k=i+1}^L w_k \right)}_{\text{backward term}} \underbrace{\left( \prod_{m=1}^{i-1} w_m \right)}_{\text{forward term}} x_0$
ResNet		$y_R = \prod_{i=1}^L (1 + w_i) x_0$	$\frac{\partial y_R}{\partial w_i} = \underbrace{\left( \prod_{k=i+1}^L (1 + w_k) \right)}_{\text{backward term}} \underbrace{\left( \prod_{m=1}^{i-1} (1 + w_m) \right)}_{\text{forward term}} x_0$
ACN		$y_A = \left( 1 + \sum_{i=1}^L \prod_{j=1}^i w_j \right) x_0$	$\frac{\partial y_A}{\partial w_i} = \underbrace{\left( 1 + \sum_{j=i+1}^L \prod_{k=i+1}^j w_k \right)}_{\text{backward term}} \underbrace{\left( \prod_{m=1}^{i-1} w_m \right)}_{\text{forward term}} x_0$

Table 1: Connectivity, Forward and Backward Propagation for FFN, ResNet, and ACN architectures.

Historically, since the emergence of Highway Networks (Srivastava et al., 2015), which first proposed additive skip connections researchers have explored numerous architectural variations. Residual Networks (ResNets) (He et al., 2016) removed learned gating functions and adopted direct identity skip connections becoming the industry standard. DenseNets (Huang et al., 2017b) utilized feature concatenation instead of addition, while FractalNets (Larsson et al., 2016) introduced a recursive tree-like architecture combining subnetworks of multiple depths to further enrich feature fusion. More recent works include learned weighted averaging across layer outputs (Pagliardini et al., 2024), application of attention mechanisms across block outputs (ElNokrashy et al., 2022) and denser connectivity patterns between network nodes (Zhu et al., 2025). Other works have explored adding scalars to either the residual or block stream to improve performance, training stability and representation learning (Savarese et al., 2016; Bachlechner et al., 2021; Zhang et al., 2024; Fischer et al., 2023). In neural machine translation, researchers have drawn inspiration from both vision and language domains to combine information from different layers, enabling richer semantic and spatial propagation throughout the network (Dou et al., 2018; Yu et al., 2018).

While Residual Networks, as discussed, offer numerous advantages—such as more robust training and improved generalization—there remain several aspects that are worth some further examination and discussion. As highlighted in (Veit et al., 2016), these architectures exhibit a notable resilience to layer dropping and permutation. In (Huang et al., 2016), it was further observed that dropping subsets of layers during training can reduce overfitting and improve generalization. In a related study, (Alain & Bengio, 2016) showed that introducing skip connections between layers can lead to parts of the network being effectively bypassed and under-trained. More recently, research has revealed substantial parameter redundancy in large-scale foundation models, particularly within their deeper layers (e.g., (Gromov et al., 2024)). All these observations can be unified under the perspective that, although residual architectures facilitate training via multiple signal pathways, these same pathways can sometimes act as shortcuts that cause certain components to be either underutilized or prone to overfitting—ultimately limiting effective generalization. Supporting this concern, (Zhang et al., 2024) demonstrated that unscaled residual connections can degrade the quality of generative representation learning, offering a concrete case where standard (unscaled) residual connections negatively impact performance. Thus, an open question remains: can we design in a principled way alternative architectures that retain the key benefits of Residual Networks—such as multiple signal pathways and efficient gradient flow—while mitigating drawbacks such as potential redundancy and shortcut overuse, effectively resulting in better representation learning?

---

Despite the breadth of the discussed research exploring different connectivity patterns across domains—with goals ranging from improved expressivity and representation learning to increased training efficiency—none of these potential improvements have achieved broad adoption beyond standard residual connections. In this work, moving a step towards answering the above questions, we explore an architectural variant where additive long feedforward connections from each layer to the output replace traditional short residual connections as shown in Table 1, introducing Auto-Compressing Networks (ACNs). ACNs showcase a unique property we coin as *auto-compression*—the ability of a network to organically compress information during training with gradient descent, through architectural design alone, dynamically pushing information to bottom layers, enhancing their representational quality, and naturally revealing redundant in deeper layers. We theoretically investigate the emergence of this property by analyzing the gradient dynamics of networks with different connectivity patterns. As illustrated in Figure 1, ACNs demonstrate layer-wise training patterns in which early layers receive significantly stronger gradients during the initial stages of training, in contrast to the more uniform gradient distribution observed in Residual Networks. Next, we empirically demonstrate a broad range of advantages that ACN-learned representations offer compared to residual or feedforward architectures, including: enhanced information compression, superior generalization, reduced catastrophic forgetting, and efficient transferability. Our contributions can be summarized as:

- We propose a novel architecture, auto-compressing networks (ACNs), that performs auto-compression organically through architectural design, addressing parameter redundancy in deep, overparameterized neural networks—a prevalent issue in modern architectures.
- We provide a detailed analysis of the gradient dynamics of ACNs, along with residual and feedforward networks, shedding light on their distinct behaviors and arguing that different connectivity patterns result in distinct learned representations.
- We implement ACNs in fully connected and transformer-based architectures and find experimentally that they achieve similar or superior performance compared to residual baselines, while 30-80% of the top layers become effective identity mappings, as all relevant information is concentrated in the bottom layers. We highlight that this approach is practical with current hardware and does not require specialized software.
- We show that ACNs learn representations that are more robust against noise and generalize better in low-data regimes compared to residual architectures.
- We argue that auto-compression offers a natural pathway to continual learning by preserving unused parameters for new tasks and utilizing different parameters for different tasks. We empirically validate this by showing that ACNs reduce catastrophic forgetting by up to 18% compared to residual networks in continual learning by preserving capacity for new unseen tasks.
- We demonstrate that ACNs outperform regularization-based approaches (relying on intermediate losses) at generalization and transfer learning without requiring hyperparameter tuning.
- We pair ACNs with widely-used baseline pruning techniques, demonstrating that their organically compressed representations significantly amplify the effectiveness of traditional compression methods, achieving superior levels of sparsity compared to residual architectures.
- We show how long-connection architectures naturally integrate feedback mechanisms, layer-wise training and developmental pruning aspects, potentially offering computational models that could inform our understanding of neurocognitive information processing while advancing more efficient artificial neural networks.

## 2 Auto-Compressing Networks

The core idea behind ACNs <sup>1</sup> is to force each layer to produce features that are directly useful for prediction. In this manner, when the last layers are pruned, earlier layers can be used for prediction directly without

---

<sup>1</sup>Code for the paper is available [here](#).

the need for further fine-tuning. Concretely, we propose replacing the residual short connections with long connections, as described in Eq. 1 and shown in Table 1 for a network of depth  $L^2$ :

$$x_i = f_i(x_{i-1}), \quad y = \sum_{i=0}^L x_i \quad (1)$$

In ACNs the output of each layer <sup>3</sup> is directly connected to the output of the network, and thus is directly optimized by the objective function during gradient descent training. Furthermore, the number of possible shortcuts is equal to the number of layers  $L$ . We find this simplification maintains the improved signal flow that shortcut connections provide, while also introducing the ability to detect potential parameter redundancy in the architecture<sup>4</sup>. We note that ACNs differ structurally from other models employing long connections, such as DenseNets (Huang et al., 2017b) and DenseFormer (Pagliardini et al., 2024), which are residual networks variants. These two models connect each layer to all preceding layers whereas ACNs connect each layer only to the output, leading to a distinct structural design<sup>5</sup>.

## 2.1 Gradient Propagation Across Network Architectures

To understand how different neural network architectures behave during training, we analyze their gradient flow characteristics. In this section, we examine and compare the forward and backward pass (gradient flow) dynamics of three architectures: traditional feedforward networks (FFN), residual networks (ResNet), and the proposed auto-compressing networks (ACN), based on the equations of Table 1. See Appendix B for a detailed derivation of the gradient equations for 1D linear neural networks.

**Notation:**  $x_i$  is the output of layer  $i$ ,  $w_i$  is the weight of layer  $i$  (the weight used to construct  $x_i$ ),  $x_0$  is the input (after a potential initial embedding operation) and  $y_F, y_R, y_A$  is the output for each architecture.

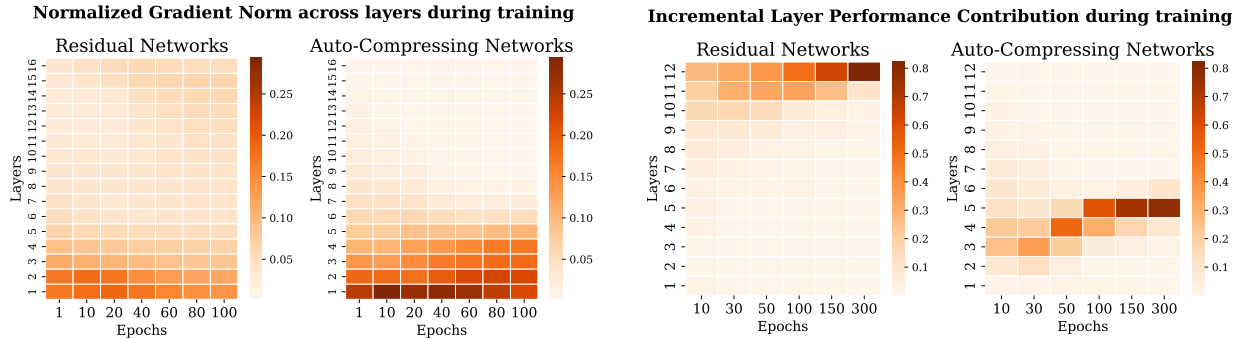


Figure 1: **(left)** ACNs vs Residual Networks gradient flow across layers during training for ViT architecture (Dosovitskiy, 2020) on ImageNet-1K, showcasing implicit layer-wise training and information concentration on the bottom layers for ACNs. On the other hand, Residual Networks show higher gradient norms (information concentration) in early and deep layers, while middle layers receive significantly lower gradients (suggesting potential parameter redundancy). **(right)** ACNs vs Residual Networks incremental performance contribution across layers during training for MLP-Mixer architecture (Tolstikhin et al., 2021) on CIFAR-10 Krizhevsky (2009), revealing auto-compression by gradual layer-wise training in ACNs (task-learning starts from shallow layers and gets "pushed" to deeper layers to maximize performance). In Residual Networks, as shown in the Figure task learning happens in the 2-3 final layers.

<sup>2</sup>We note that a classification head can be built on top of  $y$ .

<sup>3</sup>Also the embedded input, represented with  $x_0$  in equation 1.

<sup>4</sup>A careful reader may observe that long connections are a strict subset of the  $2^L$  shortcut connections in residual networks.

<sup>5</sup>We mention more details about these models in Section 10

## 2.2 Emergent Gradient Paths

**The *forward* and *backward* components:** As shown in Table 1, each gradient  $w_i$  (see  $\frac{\partial y_*}{\partial w_i}$  in the 3rd column of the respective table) decomposes into forward and backward terms. The *forward* term consists of the forward propagated signal up to layer  $i$  and determines gradient and forward propagation stability (whether the signal vanishes or explodes), while the *backward* term influences learning, containing information coming from the loss and traversing subsequent layers. For backward paths, 1D FFNs contain a single path, while 1D ACNs have  $L - i + 1$ : one direct path using the layer’s own long connection to the output, plus  $L - i$  additional paths where gradient flows from each subsequent layer’s long connection and back through the network. 1D ResNets have  $2^{L-i}$  paths since at each layer there are two options: flow through the network or follow the residual connection. It is also worth observing that ACNs feature a forward term identical to FFNs for intermediate layers (single path), while their backward components is closer to ResNets since it consists of multiple paths. Finally, it is worth mentioning that the FFN path is a subset of the ACN paths, which in turn are contained in the ResNet paths, so, for the set of paths  $B$  of the backward term one may write:

$$B_{FFN} \subset B_{ACN} \subset B_{ResNet}. \quad (2)$$

**Decomposition of the *backward* term:** The backward component (Full Gradient - **FG**) can be further decomposed into a *Network-mediated Gradient* (**NG**) component that is scaled by network weights and backpropagates information through (a subset of) the network and a *Direct Gradient* (**DG**) component that directly connects from the output to each layer, shown as the term "1" in the backpropagation equations of ACNs and ResNets in Table 1. This direct path<sup>6</sup> acts as an information super highway, especially early in training where weights are typically initialized close to zero, informing each layer directly how to contribute towards lowering the optimization objective. Finally, the **DG** contribution is more significant for ACNs compared to ResNets, due to ACNs’ linear (rather than exponential) total gradient path count.

Unlike the symmetric forward and backward terms of ResNets and FFNs, ACNs gradients, as argued, consist of a single forward path and multiple backward paths. This design creates an implicit layer-wise training dynamic, where deeper layers are trained at a slower rate compared to earlier layers, since they have a **weaker forward component** (assuming close-to-zero initialization) and a **smaller number of backward paths**. Further, when compared to ResNets, ACNs have a stronger contribution during backpropagations from the **DG** path (vs. **NG**) and this effect becomes more pronounced for deeper networks and for the early layers. For example, when training the second layer of a 1D  $L = 12$  layer network, **DG** is one of 11 ACN backward paths, while for ResNets the **DG** is competing with another 127 paths (of the **NG** term). This further accelerates training of the early layers.

**Main Claim:** We postulate that: 1) a strong **DG** component coupled with a weaker feed-forward signal leads implicitly to efficient layer-wise training, and 2) architecturally-induced layer-wise training results inadvertently in a form of **structural learning** where information is naturally pushed to early layers, i.e., later layers will become redundant (effectively identity mappings) if the earlier layers can already solve for the task. We refer to this new class of networks as **auto-compressors** since they naturally “shed” their redundant layers during backpropagation simply via architectural design. These claims are experimentally validated in the rest of the paper.

## 2.3 A Toy Demonstration

Following the theoretical analysis of gradient dynamics, we validate our key claims next through a series of experiments. First, to demonstrate how ACNs naturally compress information into early layers during training, we perform a simple toy experiment involving a 1D linear feedforward network with three layers and weights  $w_1$ ,  $w_2$  and  $w_3$  for each layer, respectively. The dataset comprises pairs drawn from the function  $y = 2x$ . We consider two architectures: the first employs residual (short) connections, while the second utilizes long connections (ACN). For this toy problem, both the residual and ACN networks should ideally

<sup>6</sup>This backward path has been previously explored as an alternative to traditional backpropagation and is typically referred to as Direct Feedback Alignment (DFA) in the literature (Nøkland, 2016; Refinetti et al., 2021).

require only a single layer to successfully accomplish the task, i.e.,  $w_1 = 1$ ,  $w_2 = 0$ ,  $w_3 = 0$  is the most efficient solution:

$$\text{ResNet: } \hat{y}_R = (1 + w_1)(1 + w_2)(1 + w_3)x = 2x, \quad (3)$$

$$\text{ACN: } \hat{y}_A = (1 + w_1 + w_1w_2 + w_1w_2w_3)x = 2x. \quad (4)$$

We perform this simple training experiment multiple times ( $N = 1000$ ), each time generating 1000 examples of input  $x$  with values between  $-10$  and  $10$ . Both models are trained for 300 epochs and the weights are initialized with values around zero either uniformly in  $[-1, 1]$  or following a normal distribution. Fig. 2 illustrates the distribution of learned  $w_1$  values for both architectures.

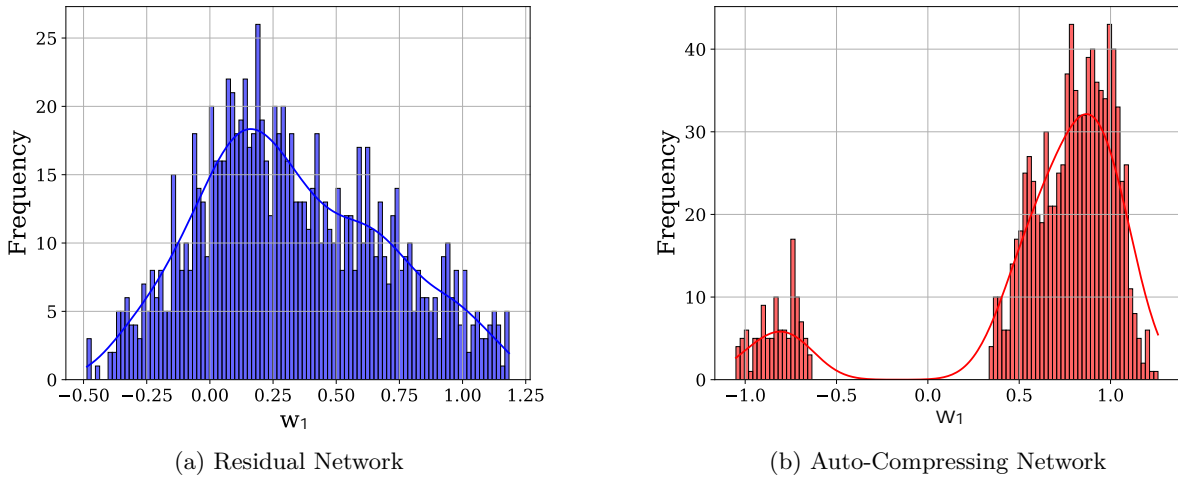


Figure 2: Histogram (1000 runs) of the weight of the first layer  $w_1$  of a 1D linear feedforward residual network with three layers solving  $y = 2x$ , utilizing either: (a) residual connections or (b) long connections (ACN).

For residual architectures in (a) we observe a pretty wide distribution of  $w_1$  values centered around 0.26; indeed  $w_1 = w_2 = w_3 \approx 0.26$  is a valid solution of Eq. 3. Thus, in this example, *residual networks have the tendency to utilize all layers equally*, even if a sparse solution exists. ACNs, however, typically converge to solutions where  $w_1$  is close to 1, allowing correct predictions from the first layer<sup>7</sup>. So in this simple example, *long connections in ACNs induce implicit depth regularization*, guiding the network toward sparse solutions.

The careful reader will observe that this behavior arises from the weight asymmetry in Eq. 4, where there are three terms that include  $w_1$ , two terms with  $w_2$  and a single term with  $w_3$ . In terms of derivative flow, the **DG** component for each layer is pretty strong compared to **NG**. Compare this with the weight symmetric Eq. 3, where the **NG** component dominates. The residual network naturally converges to the  $w_1 = w_2 = w_3$  symmetric solution<sup>8</sup>.

### 3 ACNs in Practice: Information Compression and Gradient Flow

In this section, we move from theoretical analysis and simple demonstrations to implementing auto-compressing networks in modern deep learning architectures. We apply our approach to state-of-the-art neural network

<sup>7</sup>Note that initialization plays a crucial role, as  $w_1$  sometimes converges near  $-1$  for ACNs. Further, the solution that ACN converges at is  $w_1 \approx 0.9$ ,  $w_2 \approx 0.11$ ,  $w_3 \approx 0$ , so it just gets pretty close to the most efficient solution but it does not achieve perfect compression.

<sup>8</sup>Another way to interpret Eq. 4 is that we have superimposed four feedforward networks: the identity network, a single layer, two layer and three layer network. The important tweak here is that their weights are tied, e.g.,  $w_1$  is common for all network depths, which biases the network towards a shallow solution.

models across diverse tasks and datasets, demonstrating that the information compression effect observed in our theoretical analysis manifests consistently in practice.

**Experimental Setup:** Our experimental validation spans multiple domains and model architectures. We implement ACNs using variants of the Transformer (Vaswani, 2017) for language and vision tasks and MLP-Mixer (Tolstikhin et al., 2021) for vision tasks. This allows us to evaluate our approach on diverse benchmarks including image classification (CIFAR-10, ImageNet-1K), sentiment analysis, and language understanding. In each implementation, we follow the core ACN design principle: for each input token, we compute a final output vector  $y^t$  (where  $t$  is the sequence index) by summing the output representations of all intermediate layers along with the input embedding, as shown in Eq. 1. To generate classification predictions, we either apply a pooling layer to these vectors for image classification or use the final representation of the [class] ([CLS]) token for text classification. For a network of depth  $L$ , making predictions using  $k$  intermediate layers involves computing  $y_k^t$  for each token, which is the sum of intermediate representations **up to layer  $k$** . This summed representation is then passed to a single global classification head, which is trained once and shared across all sub-networks (we do not retrain or create separate classification heads for each depth configuration). This approach yields  $L + 1$  sub-networks, ranging from using only the input embedding (the first sub-network) to the full network (the last sub-network). For example, the network shown in Figure 11(c) would be the  $L - 2$  subnetwork (utilizing layers 1 to  $L - 2$ ), whereas the network in in Figure 11(b) would be the full network (all layers included). When evaluating residual network baselines, we follow standard practice: to assess the network at depth  $k$ , we simply take the output  $y_k^t$  of the  $k$ th layer as our representation. This provides a natural comparison point to ACNs at equivalent depths. All other procedures remain the same. In all figures, prediction layer 0 refers to the input embedding passed through the classification head for prediction. Additional experimental details and hyperparameters are provided in Appendix A.

### 3.1 Auto-Compression via Direct Gradient Flow and Layer-wise training

Our experiments begin by empirically validating the main claim established in the previous section, i.e., the presence of a strong **DG** component coupled with implicit layer-wise training dynamics drives auto-compression, a property that resembles a form of structural (layer-wise) learning.

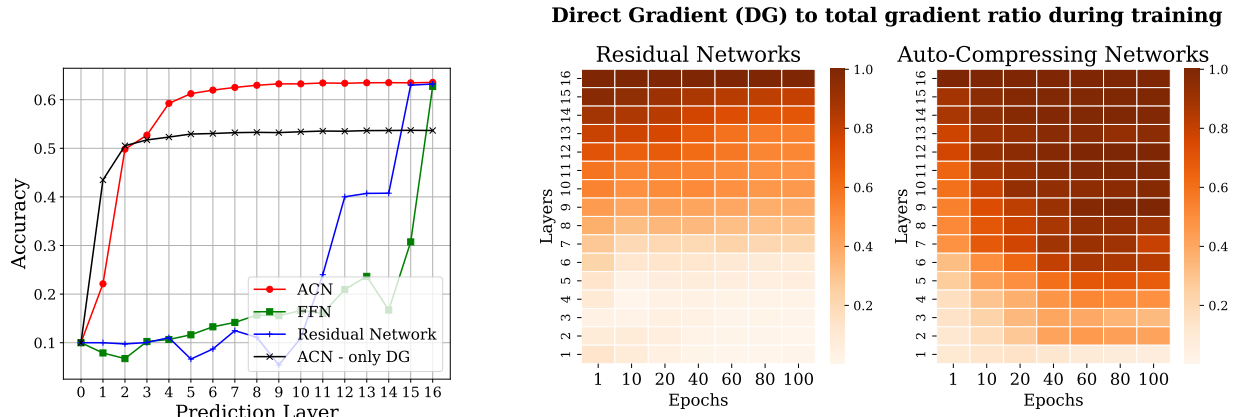


Figure 3: **(left)** ACNs is the only architectural variant that achieve auto-compression. **(right)** The ratio of direct gradient **DG** to the total gradient **FG** in auto-compressing vs residual architectures. The exponential (vs linear) number of paths in Residual Networks decreases the influence of **DG** in the training dynamics of the network compared to Auto-Compressing Networks.

To this end, We train feedforward (FFN), residual and auto-compressing variants incorporated in the MLP-Mixer architecture on CIFAR-10 dataset for 100 epochs. To emphasize the role of the DG gradient in auto-compression, we also train an ACN variant receiving gradients only from the long connections (ACN - only DG component). In Figure 3(left), we show classification accuracy plotted against network depth (layer

probing) and observe that among ACNs, FFNs, and Residual Networks, only ACNs exhibit auto-compression. Moreover, ACNs utilizing only the direct gradient (**DG**) still achieve significant auto-compression, highlighting the importance of a strong **DG** component to achieve this behavior<sup>9</sup> and explaining why FFNs do not exhibit auto-compression, as they lack a direct gradient term (Equation 7). In the case of Residual Networks, we previously argued that the exponential number of gradient paths substantially diminishes the influence of the direct gradient (**DG**) on the overall gradient, a component crucial for auto-compression. To further illustrate this, Figure 3(right) presents the ratio of **DG** to the full gradient **FG** across layers during training for both AC and Residual variants. The results indicate a significantly higher **DG** to **FG** ratio in ACNs, confirming the increased contribution of direct gradients in the early layers of auto-compressing architectures compared to residual networks and explaining the auto-compression property. Furthermore, from Figure 1 we observe that ACNs demonstrate a concentrated gradient pattern with stronger signals in early layers and stronger patterns of **layer-wise learning**. Residual Networks exhibit a more "uniform layer learning" pattern, whereas deeper layers show increasing gradient contribution in later epochs, suggesting task-specific adaptation as training progresses. Interestingly, the pattern observed in Residual Networks indicates that high gradient norms are primarily concentrated in the early and deep layers, while middle layers receive significantly lower gradients, suggesting potential redundancy.

## 4 Compression Capabilities of Auto-Compressing Networks

### 4.1 Auto-Compressing Vision Transformers

Next, we evaluate ACNs in the context of transformer architectures by implementing an auto-compressing variant of Vision Transformer (ViT) (Dosovitskiy, 2020). We train a Vision Transformer (ViT) with long connections (AC-ViT) from scratch on the ILSVRC-2012 ImageNet-1K, following the training setup in the original paper. For both models we use 256 batch size due to memory constraints. AC-ViT converges at 700 epochs, while the Residual ViT converges at 300 epochs<sup>10</sup>. As shown in Fig. 4, AC-ViT reaches top performance at only 6 layers while the vanilla ViT needs all 12 layers to reach similar performance, effectively suggesting that *ACNs can improve inference time and memory consumption without sacrificing performance*. To gain more intuition about the training dynamics and task learning of the two variants, in Figure 1(right) we plot the incremental layer performance contribution (difference in accuracy of subnetwork  $i+1$  to subnetwork  $i$ ) to track the behavior of intermediate layers throughout training. The key observation is that ACNs (right) are trained in a layer-wise fashion where early layers are trained at a faster rate and task-relevant information is gradually pushed only to a subset of the deeper layers, achieving strong performance along with auto-compression. In the contrary, the Residual variant performs task-learning in the last 2-3 layers, effectively utilizing the full network to achieve top performance.

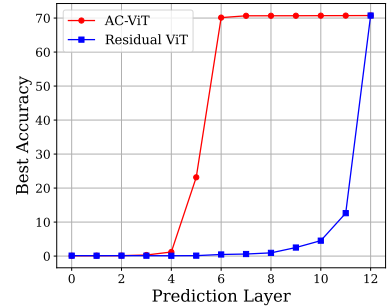


Figure 4: Performance of intermediate layers of AC vs Residual Vision Transformers trained on Imagenet-1K.

### 4.2 The Effect of Task Difficulty

Intuitively, overparameterized networks trained on easier tasks should demonstrate higher levels of redundancy. Therefore, ACNs should converge to utilizing fewer layers as task difficulty decreases. To verify this, we use the number of classes as a proxy for task difficulty for image classification on the CIFAR-10 dataset (Krizhevsky, 2009). Specifically, we create subsets of 2, 5, and 10 classes, the assumption being that binary classification should be easier than 10-class classification. For this experiment we utilize MLP-Mixer (Tolstikhin et al., 2021) and train two variants, the original MLP-Mixer with residual connections and the modified MLP-Mixer with long connections (AC-Mixer). Results are presented in Fig 5. We observe that indeed *AC-Mixer converges*

<sup>9</sup>ACNs with only the **DG** component under-perform, underpinning the importance of the **NG** component for maximizing performance.

<sup>10</sup>In this more challenging setting, we observe a trade-off between training and inference time, which is partially alleviated using a parameterization similar to DiracNets (Zagoruyko & Komodakis, 2017) for the MLP layers, specifically  $\hat{W} = (I + W)$ .



to solutions with larger effective depth, as the task “difficulty” increases. Specifically, in this experiment, ACN needs 8, 10 and 12 layers for the 2, 5 and 10-class classification problem, respectively. In contrast, the Residual Mixer converges to solutions where the full depth of the network is utilized, irrespective of the task difficulty <sup>11</sup>.

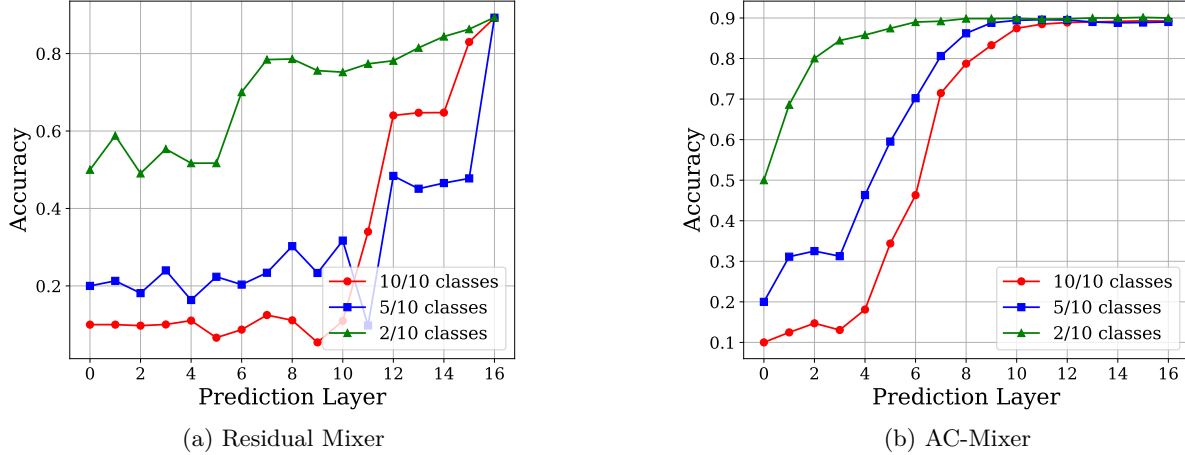


Figure 5: Performance of the intermediate layers as the number of classes (and examples) in the CIFAR-10 dataset increases from 2, to 5 to 10 classes: (a) Residual Mixer vs (b) AC-mixer.

## 5 Generalization Capabilities of Auto-Compressing Networks

While ACNs demonstrate effective parameter reduction through architectural compression, a key question remains: do these compressed representations offer additional benefits beyond parameter efficiency? In this section, we investigate whether the concentrated information in ACNs’ early layers leads to improved generalization capabilities compared to traditional residual architectures. Specifically, we explore two critical aspects of generalization: robustness to input noise and performance in low-data regimes.

### 5.1 Robustness to Input Noise

Next, we present results assessing the robustness of ACNs versus residual transformer architectures to input noise. The experiments are performed with the AC-ViT and residual ViT architectures trained on ImageNet-1K. In this experiment, we inject increasing levels of additive Gaussian noise with standard deviation  $\sigma = 0.1, 0.2, 0.4$ , and salt-and-pepper noise with percentage of altered pixels  $p = 1\%, 2\%, 10\%$ . Results (average accuracy) are shown in Table 2 (a) for Gaussian and (b) for salt-and-pepper noise. We observe that *ACNs display improved robustness to noise*, and the performance gap with the residual transformer increases as the noise levels increase. These results align with the findings of Yang et al. (2020), who showed that architectures with forward passes closer to feedforward networks (like our ACNs) exhibit enhanced noise robustness. In residual architectures, short connections allow noise to propagate and accumulate throughout the network, whereas the long-connection design of ACNs helps mitigate this amplification effect.

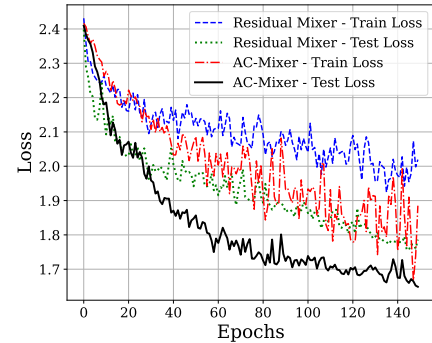


Figure 6: Train and Test Loss of AC-Mixer and Residual Mixer on CIFAR-10 (100 samples per class).

<sup>11</sup>The Residual Mixer was trained for 300 epochs, while AC-Mixer for 420 epochs to reach the performance of its residual counterpart.

Model	Baseline	Gaussian Noise			Salt and Pepper Noise		
	w/o noise	$\sigma = 0.1$	$\sigma = 0.2$	$\sigma = 0.4$	$p = 0.01$	$p = 0.05$	$p = 0.1$
Residual ViT	70.74	67.68	62.80	45.46	56.80	27.48	10.34
AC-ViT	70.76	69.50	64.54	51.89	59.80	36.35	19.98

Table 2: Robustness (average accuracy %) of ViT with long connections (AC-ViT) and with residual connections (Residual ViT) to additive Gaussian noise and salt-and-pepper noise on ImageNet-1K test set.

## 5.2 Robustness to Data Sparsity

Next, we experimentally compare the performance of residual and long connections architectures in low-data scenarios. For this purpose, we create a random subset of CIFAR-10 (Krizhevsky, 2009) by retaining only 100 samples per class, resulting in a total of 1000 examples. Using the same training settings and models as described in Section 4.2, we train both architectures for 150 epochs to assess how fast the training and test loss decrease, as a proxy for the generalization capabilities of each architecture. Results shown in Fig. 6 reveal that ACNs achieve lower training and test loss in fewer epochs compared to residual networks. This faster convergence in loss metrics is a strong indication that auto-compressing networks can be effectively utilized in scenarios with limited data.

# 6 Auto-Compressing Encoder Architectures for Language Modeling

Recent studies have demonstrated substantial parameter redundancy in modern foundation models, particularly in their deeper layers (eg. Gromov et al. (2024)). This characteristic is crucial today, in the context of large language and multimodal models, which are typically pre-trained as general-purpose models before being adapted to specific downstream tasks. Since these specialized applications may not require the full parameter capacity of the base model, learned representations (through architectural choices) that facilitate subsequent compression and pruning become crucial. In this section, we conduct a preliminary study on the effectiveness of the ACN architecture in general pre-training (masked language modeling with a BERT architecture) followed by fine-tuning and pruning. The results show that ACNs learn compact representations that: 1) achieve on-par performance with the residual architecture on transfer learning tasks, while utilizing significantly fewer parameters, and 2) complement post-training pruning techniques, enhancing their effectiveness.

## 6.1 Masked Language Modeling and Transfer Learning with ACNs

Next, we compare the ACN and residual architectures in the standard BERT pre-training and fine-tuning paradigm. Using the original BERT pretraining corpus (BooksCorpus (Zhu et al., 2015) and English Wikipedia), we train both architectures to equivalent loss values; the AC-BERT variant requires two epochs vs one epoch for the residual baseline. Following pre-training, we fine-tune both models on three GLUE benchmark datasets (Wang et al., 2018a): SST-2 sentiment analysis (Socher et al., 2013), QQP paraphrasing, and QNLI question answering (Rajpurkar et al., 2016).

Figure 7(left) demonstrates a key advantage of the ACN architecture: it naturally converges to using significantly fewer layers (approximately 75% less layers) while maintaining performance comparable to the full residual network. These results suggest promising applications for ACNs in large language models, where pre-training could be performed with long connections, allowing downstream tasks to adaptively utilize only the necessary subset of layers during fine-tuning.

## 6.2 Post-Training Pruning with AC-Encoders

ACN’s primary advantage lies in its inherent compression capabilities during training, suggesting that when combined with pruning techniques, it should significantly outperform traditional residual architectures. To provide validation for this hypothesis, we conducted experiments using magnitude and movement pruning (Sanh et al., 2020), two commonly employed baseline pruning techniques. Results are shown when fine-

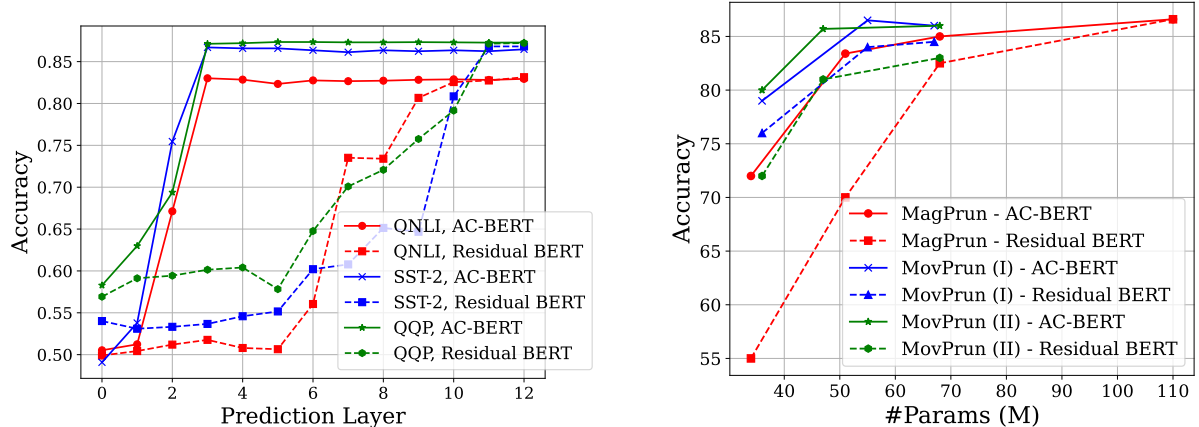


Figure 7: **(left)** Downstream performance of AC-BERT vs residual BERT on three GLUE tasks: sentiment analysis (SST-2), paraphrasing (QQP), and question answering (QNLI). **(right)** Accuracy vs model size of AC-BERT and Residual BERT on SST-2 when pruned with Magnitude and Movement Pruning (with two different settings, refer to Appendix A for details).

tuning of the SST-2 dataset sentiment analysis task. We refer to Appendix A for details regarding the experimental setup. Figure 7(right) confirms our hypothesis: ACNs consistently demonstrate superior compression-performance trade-offs compared to standard architectures, with their advantage becoming more pronounced at higher compression rates. This indicates that ACNs’ architectural design naturally leads to more efficient parameter utilization, creating representations that are inherently more amenable to further pruning. While these preliminary results validate our approach to addressing parameter redundancy, they also point toward promising future directions. We anticipate that combining pre-trained ACN architectures with state-of-the-art pruning methods will result in extremely efficient, high-performing models, though rigorous validation of this hypothesis requires further investigation.

## 7 Mitigating Catastrophic Forgetting with ACNs

Continual learning involves training models on a sequence of tasks without access to past data, aiming to retain performance on previous tasks while learning new ones (De Lange et al., 2021; Wang et al., 2024). A central challenge in CL is catastrophic forgetting—the tendency of neural networks to overwrite old knowledge when updated with new data. Common approaches include data replay methods (Rebuffi et al., 2017; Lopez-Paz & Ranzato, 2017) and regularization techniques that penalize changes to important parameters (Kirkpatrick et al., 2017; Zenke et al., 2017; Aljundi et al., 2018). We’ve already demonstrated that ACNs, through implicit layer-wise training, dynamically allocate parameters based on task demands while preserving redundant parameters for future tasks. Conversely, Residual Networks optimized for efficient task learning risk overfitting and suboptimal parameter usage in these sequential learning settings. To test our claims, we evaluate both architectures on the split CIFAR-100 continual learning benchmark, comprising 20 sequential disjoint 5-class classification tasks, focusing on task-incremental learning (Van de Ven & Tolias, 2019) where task identity is known. We utilize MLP-Mixer architectures (hyperparameters in Appendix A) and we test two continual learning algorithms trained for 10 epochs for each task: naive fine-tuning (Naive FT) and Synaptic Intelligence (SI) (Zenke et al., 2017), which adds a gradient-based regularizer to each parameter depending on how changes in it affect the total loss in a task over the training trajectory. Across experiments, we report Average Forgetting, defined as the mean difference between a task’s best performance (right after it is learned) and its final performance after all tasks are learned, and Average Accuracy, defined as the mean accuracy over all tasks at the end of training. We expect gradient-based regularization methods to perform particularly well with ACNs since unused, redundant parameters receive small gradients, making their detection easier compared to Residual Networks where gradients are more uniformly distributed (see gradient heatmaps, Fig. 1(left)). Results in Table 3 confirm our intuition: ACNs consistently exhibit significantly less forgetting (up to 18% improvement) compared to Residual Networks. Notably, with SI, increasing ACN

depth decreases forgetting—an ideal behavior for CL systems where increasing network capacity reduces forgetting—while Residual Networks show the opposite pattern, indicating potential overfitting. ACNs also achieve better average accuracy across all tasks, further establishing them as a more suitable architecture for continual learning.

Method	Arch	Avg. Accuracy (%) $\uparrow$			Avg. Forgetting (%) $\downarrow$		
		$L = 5$	$L = 10$	$L = 15$	$L = 5$	$L = 10$	$L = 15$
Naive FT	AC-Mixer	$32.97 \pm 2.4$	$32.94 \pm 5.3$	$31.61 \pm 2.2$	$46.55 \pm 2.2$	$45.46 \pm 5.8$	$46.91 \pm 2.4$
	ResMixer	$31.77 \pm 1.8$	$28.16 \pm 1$	$26.14 \pm 2.3$	$52.76 \pm 2.3$	$54.89 \pm 1.6$	$54.49 \pm 2.2$
SI	AC-Mixer	$44.5 \pm 2.2$	$46.1 \pm 1.3$	<b><math>46.2 \pm 0.8</math></b>	$35.7 \pm 2.1$	$33.8 \pm 0.4$	<b><math>32 \pm 1.8</math></b>
	ResMixer	$43.47 \pm 3.1$	$36.1 \pm 5$	$32.1 \pm 0.8$	$42.4 \pm 4.1$	$44.6 \pm 3.7$	$50 \pm 2.1$

Table 3: Average accuracy and forgetting across layers, methods, and architectures on the Split CIFAR-100 continual learning benchmark. Models are trained for 10 epochs per task, where each task consists of classifying 5 out of 100 classes presented sequentially.  $L$  denotes the number of layers in the architecture. ACNs consistently **forget less** and they also **do not waste capacity**.

## 8 Auto-Compressing Architectures vs. Layer-wise Loss Regularization

Parameter redundancy, and specifically potential layer redundancy, in residual architectures is a phenomenon that has been well documented (Alain & Bengio, 2016; Veit et al., 2016; Huang et al., 2016). Recent works (Elhoushi et al., 2024b; Jiang et al., 2024) have attempted to address this through regularization-based layer-wise structural learning approaches during training, specifically by adding losses to all intermediate layers of the network and using a weighted sum of them as the total loss, a technique formally introduced in (Lee et al., 2015) for improved training. Such loss-based regularization methods rely heavily on precise tuning of intermediate loss weights, creating practical challenges. If early-layer loss weights are set too high, the network risks overfitting and poor generalization; if set too low, performance improves gradually across layers with no clear cutoff point, reaching optimal results only at the final layer. This sensitivity to hyperparameter selection makes it difficult to reliably identify an optimal depth for inference using loss-based regularization. ACNs address this challenge through architectural design rather than regularization, naturally compressing information without requiring complex hyperparameter tuning.

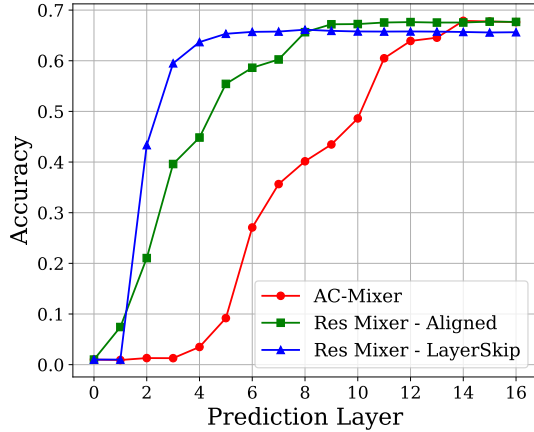
### 8.1 ACNs achieve better generalization

To evaluate hyperparameter sensitivity in regularization-based approaches, we compare several methods on the CIFAR-10 dataset using MLP-Mixer architectures. Our comparison includes: 1) our proposed AC-Mixer, 2) an unregularized Residual Mixer as baseline, 3) a Residual Mixer with the setup of (Jiang et al., 2024) (Aligned), 4) a Residual Mixer with the setup of (Elhoushi et al., 2024b) (LayerSkip), with the rotational early exit curriculum with  $p_{max} = 0.1$ ,  $e_{scale} = 0.2$  and  $C_{rot,R} = 15$ , 5) a Residual Mixer with a baseline vanilla deep supervision (Lee et al., 2015) where all intermediate losses before the final layer are weighted with  $\lambda = 0.1$  (DeepSup). This comparative analysis reveals how different approaches respond to their respective hyperparameter configurations. We follow the training pipeline as described in the previous section and track the performance of all intermediate layers. Our experiments (Figure 8) demonstrate that while regularization approaches are highly sensitive to intermediate loss weights, creating a trade-off between performance and compression, ACNs consistently achieve strong results through their inherent architectural properties. Specifically, ACNs match the performance of unregularized Residual Networks, while effectively determining a shallower cutoff layer. While careful tuning of regularization methods can potentially match ACN’s performance, our approach provides a more elegant and robust solution that requires no parameter adjustment while maintaining high performance and achieving sparsity.

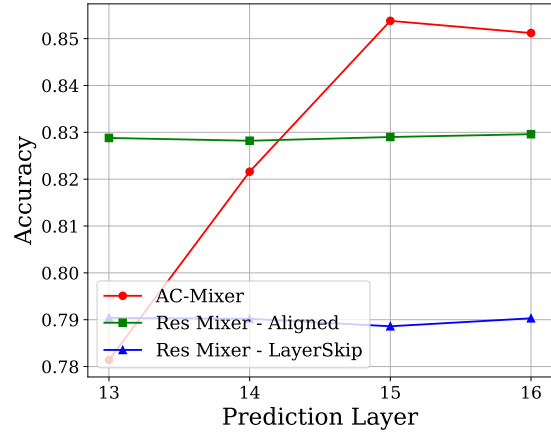
### 8.2 ACNs show stronger Transfer Learning capabilities

To evaluate whether different layer compression approaches learn generalizable representations, we conduct a transfer learning experiment from CIFAR-100 to CIFAR-10 (Krizhevsky, 2009). This setup allows us to assess how well each model’s learned representations transfer to a similar task. In regularization-based layer compression methods, explicitly training all layers to directly minimize a task loss through intermediate supervision can lead to overfitting, as shown in the previous section, which can further result in weaker transfer capabilities on downstream tasks. In contrast, ACNs’ implicit compression mechanism naturally balances generalizability and task performance without imposing external constraints.

To ensure fair comparison, we train all models to achieve comparable performance on the CIFAR-100 pre-training task, enabling direct assessment of their transfer capabilities to CIFAR-10. The results in Figure 9 confirm our analysis: ACNs demonstrate superior performance on the downstream CIFAR-10 task compared to regularization-based methods, even when upstream CIFAR-100 task performance is similar. This provides additional evidence that the representations learned by ACNs are more generalizable and thus exhibit greater transferability.



(a) C-100 performance.



(b) C-100 to C-10 Transfer.

Figure 9: Transfer learning performance (C-100 to C-10) of AC-Mixer and Residual-Mixer with intermediate losses based on (Jiang et al., 2024) (Aligned) and (Elhoushi et al., 2024b) (LayerSkip).

## 9 Summary of Results

From our experiments, we conclude that ACNs are able to “push” information down to the early neural layers without performance degradation, resulting in a sparse high-performing network, effectively revealing potential redundant layers and driving the pruning process. This pruning significantly reduces memory requirements and accelerates inference. In all of the experiments we observed that the converged depth between train and validation/test sets matched. Thus, pruning layer depth is a meta-parameter determined directly on the validation set, eliminating the need for a separate pruning procedure after training. Additionally, utilizing ACNs makes it straightforward to produce and distribute differently sized variants of the same architecture with a single training run—for example, distributing tiny, small, medium, and large versions of the model.

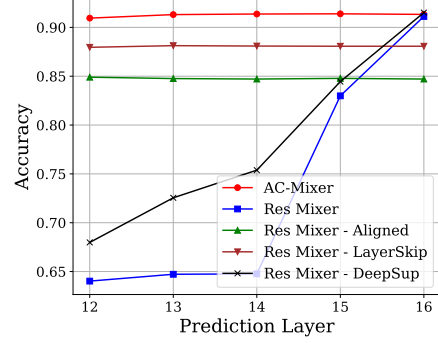


Figure 8: Performance of intermediate layers on CIFAR-10 of AC-Mixer, unregularized Residual Mixer and Residual-Mixer with intermediate losses based on (Jiang et al., 2024) (Aligned), (Elhoushi et al., 2024b) (LayerSkip) and (Lee et al., 2015) (DeepSup). For better resolution only the last 4 layers are shown in the plot.

## Performance vs. Resource Usage Trade-off

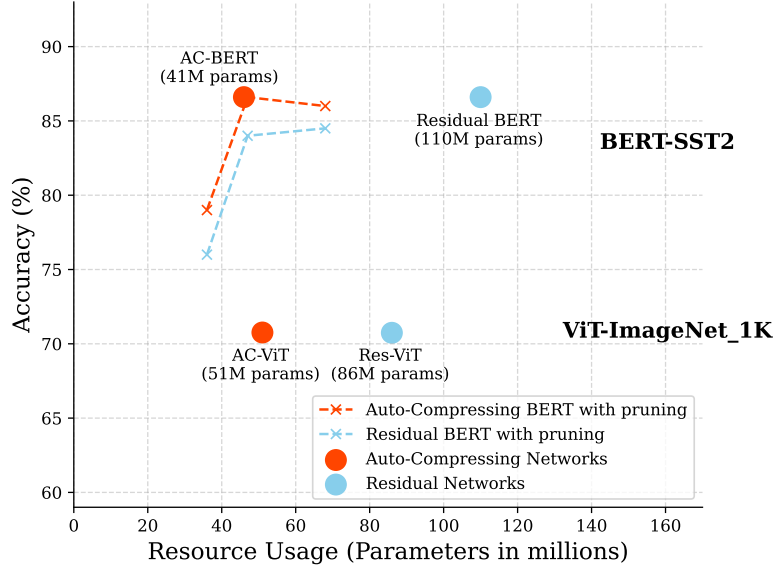


Figure 10: ACNs vs Residual Networks performance vs number of parameters, with and without pruning.

In Figure 10, we show the performance of the pruned ACN models compared to their respective residual baselines across various experiments. We also note that utilizing the pruned AC-ViT instead of the residual ViT, we can reduce inference time from 13.9 miliseconds to 8.3 miliseconds (CPU).

Beyond parameter efficiency, our experiments demonstrated several additional advantages of ACNs. In noise robustness tests, AC-ViT maintained 51.89% accuracy under severe Gaussian noise  $\sigma = 0.4$  compared to ResNet-ViT’s 45.46%, and nearly doubled performance (19.98% vs 10.34%) under heavy salt-and-pepper noise at  $p = 0.1$ . When trained on limited data (100 samples per class), ACNs converged to lower training and test losses significantly faster than residual networks. In transfer learning from CIFAR-100 to CIFAR-10, ACNs achieved 85.7% accuracy compared to 83.2% for the best regularization-based approach while using fewer parameters. In a continual learning setting, we observe that ACNs significantly reduce forgetting—by up to 18% compared to ResNets—while effectively leveraging additional model capacity (i.e., forgetting is reduced as more layers are added). Finally, when combined with pruning techniques on language tasks, ACNs maintained 80% accuracy at sparsity levels where residual models dropped to 65%, demonstrating that architectural compression and pruning techniques are complementary rather than redundant.

## 10 Background and Related Work

Next we review related work in four key areas: 1) residual connections and their role in training stability, 2) architectural variants with longer/denser residual connections, 3) methods that employ intermediate layer losses to learn better representations and exit early, and 4) neural architectures that induce regularization and better representations. Our work is more closely related to research area 4, but it is important to note that training stability, efficiency, performance, and representation learning are related goals, which can be achieved either through architectural choices (1, 2, 4) and / or through additional optimization criteria (3).

**Residual Connections and Training Stability:** Training deep neural networks with gradient descent becomes increasingly difficult as network depth increases. Multipath network architectures date back to the 1980s, with early work exploring cascade structures in fully connected networks trained layer by layer to improve training stability (Fahlman & Lebiere, 1989). Highway networks (Srivastava et al., 2015) introduced gated bypass paths that allowed for effective training of networks with hundreds of layers. (He et al., 2016) found that deeper convolutional neural networks (CNNs) not only suffer from a decrease in generalization

performance, often due to overfitting, but also experience a decrease in training performance. To address this, they introduced residual connections (or identity mappings), proposing that learning residual functions relative to identity mappings simplifies optimization. These skip connections improve the training process and often improve performance (Balduzzi et al. (2017), Orhan & Pitkow (2017), Zaeemzadeh et al. (2020), Li et al. (2018)). Veit et al. (2016) further argued that a residual network with  $n$  layers can be viewed as a collection of  $2^n$  paths of varying length. At each layer, the signal either skips the layer or passes through it, creating  $2^n$  possible paths. Despite sharing weights, these paths function as an ensemble of networks, as confirmed by experiments. In contrast, a traditional deep feedforward network has only one path, so removing any random layer significantly degrades performance. Additionally, the authors showed that these paths are typically shallow, with backward gradients often vanishing after passing through only a small fraction of the total layers.

**Residual Variants:** Following the success of residual networks various architectural modifications were proposed to improve efficiency and performance. In DenseNets (Huang et al., 2017b), each layer is connected to all subsequent layers enabling for more efficient feature reuse; fusion is achieved through concatenation rather than addition. More recently, DenseFormer (Pagliardini et al., 2024) introduced learned weighted averaging across layer outputs, while Depth-Wise Attention (ElNokrashy et al., 2022) applies attention mechanisms across block outputs.

**Intermediate Supervision and Early Exit:** In deeply supervised nets (Lee et al., 2015), complementary objectives are added to all intermediate layers to encourage hidden layers to learn more discriminative representations. In this approach, each intermediate objective  $i$  is a loss function that captures the classification error of an SVM trained on the output features of layer  $i$ . The overall loss is the sum of the intermediate and final objectives. This idea evolved in several directions: Graves (Graves, 2016) proposed adaptive computation time, while more recent work like MSDNet (Huang et al., 2017a) and CALM (Schuster et al., 2022) introduced dedicated prediction heads. Other approaches employ trainable routing mechanisms Wang et al. (2018b); Wu et al. (2018) to determine layer usage. Concurrent to our work, LayerSkip Elhoushi et al. (2024a) proposes an architecture similar to ACNs, focusing primarily on inference acceleration through layer dropout and early exit mechanisms. Additionally, (Jiang et al., 2024) also incorporates intermediate losses with a common head and a linearly increasing weight curriculum, justifying it through the lens of representational similarity between intermediate layers. While these approaches rely on explicit auxiliary objectives or dedicated components, our work achieves similar benefits through architectural design alone, enabling natural depth determination through gradient-based optimization.

**Architecturally-induced regularization and representation learning:** Stochastic regularization methods like Dropout Srivastava et al. (2014) and its variants demonstrated that randomly dropping connections during training can lead to more robust feature learning. This insight was extended to other structural approaches like Stochastic Depth Huang et al. (2016) where randomly dropping entire layers improved generalization. Residual connections initially proposed to address the vanishing gradient problem He et al. (2016) have been shown to contribute to smoother loss landscapes and improved generalization Li et al. (2018). These findings align with theoretical work showing that architectural choices impose implicit biases that influence the solutions found during training Gunasekar et al. (2018). Recent work on transformers shows that architectural choices like attention patterns and layer normalization can also induce implicit regularization effects, e.g., the combination of skip connections and layer normalization can bias the model toward low-rank solutions Bai et al. (2021). The proposed long connection approach builds on these insights, using architectural design to naturally encourage the learning of robust representation while enabling automatic information compression allowing for early exit.

## 11 Discussion

**Efficient learning mechanisms in Biological neural networks:** Biological neural networks provide valuable insights for addressing challenges like parameter allocation and efficient learning, as they seamlessly combine bottom-up entropy-based learning (extracting statistical regularities from sensory input) with top-down task-based feedback to develop remarkably efficient representations. Complementary to this feedback architecture, the brain also exhibits developmental refinement through synaptic pruning (Peter R., 1979),

---

where initial overconnectivity is followed by experience-dependent elimination of less active connections. Research suggests that this pruning process may be guided in part by feedback signals that help identify which connections are most relevant for tasks the organism frequently encounters (Changeux & Danchin, 1976; Katz & Shatz, 1996). The developmental trajectory also demonstrates a form of layer-wise maturation, as deeper cortical layers and higher cognitive regions tend to develop and refine after earlier sensory processing regions have established their basic functionality (Rakic et al., 1986; Casey et al., 2005). These complementary mechanisms –long-range feedback, activity-dependent refinement, and sequential layer maturation– work together to create neural circuits that are efficient and specifically adapted to environmental demands. In machine learning, these biological principles have inspired approaches such as greedy layer-wise pre-training (Hinton et al., 2006; Bengio et al., 2006), network pruning methods (Han et al., 2015; Frankle & Carbin, 2018; Zhu & Gupta, 2017), and feedback networks (Zamir et al., 2017; Paraskevopoulos et al., 2022), but to date, no approach has successfully integrated the feedback mechanisms and developmental trajectory aspects into a unified neural architecture.

**Biological motivation for long-connections:** Brain Neural Networks (BNNs) combine short and long connections (Bassett & Bullmore, 2006), where short connections form dense sub-network hubs, and long connections sparsely link these hubs. Typically, short connections are more numerous and have stronger synaptic weights (Muldoon et al., 2016). In (Betz et al., 2018), the authors show that short connections more efficiently route information across brain areas and sub-networks. Long connections are key for functional diversity, offering unique inputs and novel targets for outputs across sub-networks. The importance of long connections for BNNs is highlighted in imaging (Ecker et al., 2015) and computational modeling studies (McClelland, 2000) showing “evidence both of local over-connectivity and of long-distance under-connectivity” in BNNs of individuals on the autistic spectrum (Wass, 2011). This served as our main motivation for exploring long connections in search of architectures that can lead to better representations, improved generalization and enhanced performance in complex tasks.

**Biological motivation for auto-compressing networks:** The brain itself has mechanisms for creating efficient and robust biological networks. One key efficiency mechanism is synaptic pruning. During early development, an excess of synapses is formed and progressively eliminated through activity-dependent pruning (Sakai, 2020). Early studies (Peter R., 1979) measured synaptic density across different ages and found that it peaks around 1–2 years of age, followed by a decline to approximately 50% by adulthood. This approach of early overconnectivity followed by pruning has been shown to train neural networks exhibiting significant efficiency and robustness (Navlakha et al., 2015). ACNs can be viewed as an initial architectural approach to determining the essential number of layers while learning a task (experience-based), by starting from an overparameterized network at initialization and exploring the parameter space during training. Note, however, that as we have experimentally verified in Section 6.2, computational ANN pruning algorithms (motivated by BNN’s “use it or lose it” synaptic pruning) can be effectively combined with the ACN architecture to achieve even greater compression gains.

**Connection to layer-wise training:** Greedy layer-wise training (Hinton et al., 2006; Bengio et al., 2006) was a popular method for training deep neural networks in a sequential manner, inspired by cognitive neural development in the prefrontal cortex (DeFelipe, 2011). In our analysis of ACN’s dynamics, we show that ACNs naturally exhibit similar layer-wise training behavior, where early layers train first followed by deeper layers. Unlike traditional layer-wise training that requires explicitly freezing layers and careful hyperparameter tuning, this sequential training emerges automatically in ACNs due to their long-connection architecture, effectively providing a "one-shot" version of layer-wise training.

**ACNs vs ResNets connectivity patterns:** Residual Networks were motivated by improving training robustness - their skip connections were designed to facilitate gradient flow and enable stable training of networks of great depth. However, this architectural innovation had an unexpected benefit: ResNets also demonstrated better generalization compared to standard feedforward networks. This improved generalization appears to stem from the ensemble-like behavior created by the multiple paths through which information can flow. ACNs take inspiration from biological networks’ sparse but strategic connectivity patterns. By maintaining direct long-range connections to the output while reducing local skip connections, ACNs achieve both stable training and enhanced generalization through a different mechanism. Rather than relying on dense connectivity and ensemble-like behavior, ACNs encourage the development of more abstract and integrated



---

representations in earlier layers. This architectural choice appears to better support generalization while maintaining robustness in the training. A promising future direction is to integrate small-world network properties into artificial network architectures.

**ACNs vs ResNets training time:** In our experiments, we observed a trade-off between training and inference cost when choosing between short residual connections and long connections. It is possible that ACN’s inherent sparsity and longer training time is what makes them more robust to noise and more efficient in low-data settings. Preliminary experiments further strengthen this belief: as the representations learned by earlier layers become more discriminative focusing on the task at hand, ACNs can still effectively transfer their knowledge to downstream tasks.

**A Remark:** Altering connectivity patterns in artificial networks may provide valuable insights that could be mapped back to biological network structures. This approach may contribute to a deeper understanding of why evolutionary processes have led to the specific connectivity patterns observed in biological systems.

## 12 Conclusions

In this work, we introduced Auto-Compressing Networks (ACNs), an architectural design that organically compresses information into early layers of a neural network during training via long skip connections from each layer to the output, a property we coined as *auto-compression*. Unlike residual networks, ACNs do not require explicit compression objectives or regularization; instead, they leverage architectural design and gradient-based optimization to induce implicit layer-wise training dynamics that drive auto-compression.

Our theoretical and empirical analyses demonstrate that ACNs alter gradient flow, imposing implicit layer-wise training dynamics and resulting in distinct representations compared to feedforward and residual architectures. In practice, this leads to 30–80% of upper layers becoming effectively redundant, enabling faster inference and reduced memory usage without sacrificing accuracy. Experiments across diverse modalities (vision, language) and architectures (ViTs, Mixers, BERT) further show that ACNs match or outperform residual baselines, while offering greater robustness to noise and low-data regimes, excelling in transfer learning, boosting pruning techniques and reducing catastrophic forgetting by up to 18%—all without specialized tuning, overall suggesting that they learn better representations despite using fewer parameters.

Concluding, Auto-Compressing Networks (ACNs), building on implicit regularization through architectural design insights, represent a promising step towards more self-adapting neural architectures that allocate resources based on the task at hand, while learning sparse yet robust representations. Future research could expand ACNs to self-supervised and multi-task settings, leveraging the pre-training and fine-tuning paradigm. ACNs also hold promise for generative tasks, reducing inference costs and energy consumption. Additionally, developing inference-time algorithms that dynamically adjust the number of layers per sample for optimal performance and efficiency is an intriguing direction for future work. Last but not least, ACNs are only one possible long-connection architecture out of the many that are worth investigating further.

## 13 Limitations

Due to resource constraints, our proposed architecture was evaluated solely on relatively small scale tasks; however, it demonstrated robust and promising performance across various modalities, datasets, and state-of-the-art architectures within this scope. To fully assess its potential and limitations, further testing on a broader range of tasks is essential. Additionally, applying our method in self-supervised and multi-task learning settings, such as training large-scale language models or multimodal models, represents a significant and exciting avenue for future research.

Another limitation is the increased training time observed with ACNs compared to traditional architectures with residual connections. While we partially addressed this issue by employing parameterizations similar to DiracNets, a more comprehensive solution to reduce training time remains an open question. Of course this could be both a blessing and a curse, as longer training times might lead to learning better representations. In any case, further research into training schedules and initialization schemes is needed to resolve this trade-off.

---

## 14 Broader Impact

Our work contributes to the development of more efficient and robust neural network architectures by drawing inspiration from biological processes. By enabling networks to identify and prune redundant layers, we aim to reduce computational, memory, and energy requirements during inference, which will have a significant impact with broader adoption of AI technology. Conceptually, this line of research could lead to network architectures with inherent System 1 and System 2 capabilities (Kahneman, 2011), where networks adaptively use fewer layers—analogue to fast thinking—for easier tasks, and engage more layers—resembling slow thinking—for more complex tasks.

However, as with any advancement in AI, there is a potential for misuse. More efficient models could be leveraged to deploy AI systems more broadly, including in areas with insufficient oversight or in applications that may infringe on privacy or other ethical considerations.

## References

- Guillaume Alain and Yoshua Bengio. Understanding intermediate layers using linear classifier probes. *arXiv preprint arXiv:1610.01644*, 2016.
- Rahaf Aljundi, Francesca Babiloni, Mohamed Elhoseiny, Marcus Rohrbach, and Tinne Tuytelaars. Memory aware synapses: Learning what (not) to forget. In *Proceedings of the European conference on computer vision (ECCV)*, pp. 139–154, 2018.
- Thomas Bachlechner, Bodhisattwa Prasad Majumder, Henry Mao, Gary Cottrell, and Julian McAuley. Rezero is all you need: Fast convergence at large depth. In *Uncertainty in Artificial Intelligence*, pp. 1352–1361. PMLR, 2021.
- Yu Bai, J. Zico Kolter, and Vladlen Koltun. Understanding the low-rank bias of deep neural networks. In *International Conference on Learning Representations*, 2021.
- David Balduzzi, Marcus Frean, Lennox Leary, JP Lewis, Kurt Wan-Duo Ma, and Brian McWilliams. The shattered gradients problem: If resnets are the answer, then what is the question? In *International conference on machine learning*, pp. 342–350. PMLR, 2017.
- Danielle Smith Bassett and ED Bullmore. Small-world brain networks. *The neuroscientist*, 12(6):512–523, 2006.
- Yoshua Bengio, Pascal Lamblin, Dan Popovici, and Hugo Larochelle. Greedy layer-wise training of deep networks. *Advances in neural information processing systems*, 19, 2006.
- Richard F. Betzel and Danielle S. Bassett. Specificity and robustness of long-distance connections in weighted, interareal connectomes. *Proceedings of the National Academy of Sciences*, 115(21):E4880–E4889, 2018. doi: 10.1073/pnas.1720186115.
- Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners, 2020. URL <https://arxiv.org/abs/2005.14165>.
- B.J. Casey, Nim Tottenham, Conor Liston, and Sarah Durston. Imaging the developing brain: what have we learned about cognitive development? *Trends in Cognitive Sciences*, 9(3):104–110, 2005. ISSN 1364-6613. doi: <https://doi.org/10.1016/j.tics.2005.01.011>. URL <https://www.sciencedirect.com/science/article/pii/S1364661305000306>. Special issue: Developmental cognitive neuroscience.
- Jean Pierre Changeux and Antoine Danchin. Selective stabilisation of developing synapses as a mechanism for the specification of neuronal networks. *Nature*, 264:705–712, 1976. URL <https://api.semanticscholar.org/CorpusID:4241758>.

- 
- Matthias De Lange, Rahaf Aljundi, Marc Masana, Sarah Parisot, Xu Jia, Aleš Leonardis, Gregory Slabaugh, and Tinne Tuytelaars. A continual learning survey: Defying forgetting in classification tasks. *IEEE transactions on pattern analysis and machine intelligence*, 44(7):3366–3385, 2021.
- Javier DeFelipe. The evolution of the brain, the human nature of cortical circuits, and intellectual creativity. *Frontiers in Neuroanatomy*, 5, 2011. ISSN 1662-5129. doi: 10.3389/fnana.2011.00029. URL <https://www.frontiersin.org/journals/neuroanatomy/articles/10.3389/fnana.2011.00029>.
- Alexey Dosovitskiy. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.
- Zi-Yi Dou, Zhaopeng Tu, Xing Wang, Shuming Shi, and Tong Zhang. Exploiting deep representations for neural machine translation. *arXiv preprint arXiv:1810.10181*, 2018.
- Christine Ecker, Susan Y Bookheimer, and Declan G M Murphy. Neuroimaging in autism spectrum disorder: brain structure and function across the lifespan. *The Lancet Neurology*, 14(11):1121–1134, 2015. ISSN 1474-4422. doi: [https://doi.org/10.1016/S1474-4422\(15\)00050-2](https://doi.org/10.1016/S1474-4422(15)00050-2). URL <https://www.sciencedirect.com/science/article/pii/S1474442215000502>.
- Mostafa Elhoushi, Akshat Shrivastava, Diana Liskovich, Basil Hosmer, Bram Wasti, Liangzhen Lai, Anas Mahmoud, Bilge Acun, Saurabh Agarwal, Ahmed Roman, Ahmed Aly, Beidi Chen, and Carole-Jean Wu. Layerskip: Enabling early exit inference and self-speculative decoding. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 12622–12642. Association for Computational Linguistics, 2024a. doi: 10.18653/v1/2024.acl-long.681. URL <http://dx.doi.org/10.18653/v1/2024.acl-long.681>.
- Mostafa Elhoushi, Akshat Shrivastava, Diana Liskovich, Basil Hosmer, Bram Wasti, Liangzhen Lai, Anas Mahmoud, Bilge Acun, Saurabh Agarwal, Ahmed Roman, et al. Layerskip: Enabling early exit inference and self-speculative decoding. *arXiv preprint arXiv:2404.16710*, 2024b.
- Muhammad ElNokrashy, Badr AlKhamissi, and Mona Diab. Depth-wise attention (dwatt): A layer fusion method for data-efficient classification. *arXiv preprint arXiv:2209.15168*, 2022.
- Scott Fahlman and Christian Lebiere. The cascade-correlation learning architecture. *Advances in neural information processing systems*, 2, 1989.
- Kirsten Fischer, David Dahmen, and Moritz Helias. Field theory for optimal signal propagation in resnets. *arXiv preprint arXiv:2305.07715*, 2023.
- Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. *arXiv preprint arXiv:1803.03635*, 2018.
- Alex Graves. Adaptive computation time for recurrent neural networks. *CoRR*, abs/1603.08983, 2016. URL <http://arxiv.org/abs/1603.08983>.
- Andrey Gromov, Kushal Tirumala, Hassan Shapourian, Paolo Glorioso, and Daniel A Roberts. The unreasonable ineffectiveness of the deeper layers. *arXiv preprint arXiv:2403.17887*, 2024.
- Suriya Gunasekar, Jason Lee, Tong Zhang, and Behnam Neyshabur. Implicit bias of gradient descent on wide deep networks. In *Advances in Neural Information Processing Systems*, 2018.
- Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*, 2015.
- Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.
- Geoffrey E. Hinton, Simon Osindero, and Yee-Whye Teh. A fast learning algorithm for deep belief nets. *Neural Computation*, 18(7):1527–1554, 07 2006. ISSN 0899-7667. doi: 10.1162/neco.2006.18.7.1527. URL <https://doi.org/10.1162/neco.2006.18.7.1527>.

- 
- Gao Huang, Yu Sun, Zhuang Liu, Daniel Sedra, and Kilian Q Weinberger. Deep networks with stochastic depth. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part IV 14*, pp. 646–661. Springer, 2016.
- Gao Huang, Danlu Chen, Tianhong Li, Felix Wu, Laurens van der Maaten, and Kilian Q. Weinberger. Multi-scale dense networks for resource efficient image classification. In *International Conference on Learning Representations*, 2017a.
- Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4700–4708, 2017b.
- Jiachen Jiang, Jinxin Zhou, and Zhihui Zhu. Tracing representation progression: Analyzing and enhancing layer-wise similarity. In *The Thirteenth International Conference on Learning Representations*, 2024.
- Daniel Kahneman. Thinking, fast and slow. *Farrar, Straus and Giroux*, 2011.
- L. C. Katz and C. J. Shatz. Synaptic activity and the construction of cortical circuits. *Science*, 274(5290):1133–1138, 1996. doi: 10.1126/science.274.5290.1133. URL <https://www.science.org/doi/abs/10.1126/science.274.5290.1133>.
- James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017.
- Alex Krizhevsky. Learning multiple layers of features from tiny images. pp. 32–33, 2009. URL <https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf>.
- Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 2012.
- Gustav Larsson, Michael Maire, and Gregory Shakhnarovich. Fractalnet: Ultra-deep neural networks without residuals. *arXiv preprint arXiv:1605.07648*, 2016.
- Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 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.
- Hao Li, Wei Fan, Yang Zhang, Tian Yu, Yandong Chen, Steve Horvath, Romain Combes, Tong Liu, Jian Ma, Stefano Soatto, et al. Visualizing the loss landscape of neural nets. In *Advances in Neural Information Processing Systems*, 2018.
- David Lopez-Paz and Marc’Aurelio Ranzato. Gradient episodic memory for continual learning. *Advances in neural information processing systems*, 30, 2017.
- I Loshchilov. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.
- James L McClelland. The basis of hyperspecificity in autism: A preliminary suggestion based on properties of neural nets. *Journal of Autism and Developmental Disorders*, 30:497–502, 2000.
- Sarah Feldt Muldoon, Eric W Bridgeford, and Danielle S Bassett. Small-world propensity and weighted brain networks. *Scientific reports*, 6(1):22057, 2016.
- Saket Navlakha, Alison L Barth, and Ziv Bar-Joseph. Decreasing-rate pruning optimizes the construction of efficient and robust distributed networks. *PLoS computational biology*, 11(7):e1004347, 2015.
- Arild Nøkland. Direct feedback alignment provides learning in deep neural networks. *Advances in neural information processing systems*, 29, 2016.

- 
- A Emin Orhan and Xaq Pitkow. Skip connections eliminate singularities. *arXiv preprint arXiv:1701.09175*, 2017.
- Matteo Pagliardini, Amirkeivan Mohtashami, Francois Fleuret, and Martin Jaggi. Denseformer: Enhancing information flow in transformers via depth weighted averaging. *arXiv preprint arXiv:2402.02622*, 2024.
- Georgios Paraskevopoulos, Efthymios Georgiou, and Alexandras Potamianos. Mmlatch: Bottom-up top-down fusion for multimodal sentiment analysis. In *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 4573–4577. IEEE, 2022.
- Huttenlocher Peter R. Synaptic density in human frontal cortex — developmental changes and effects of aging. *Brain Research*, 163(2):195–205, 1979. ISSN 0006-8993. doi: [https://doi.org/10.1016/0006-8993\(79\)90349-4](https://doi.org/10.1016/0006-8993(79)90349-4). URL <https://www.sciencedirect.com/science/article/pii/0006899379903494>.
- Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. SQuAD: 100,000+ questions for machine comprehension of text. In Jian Su, Kevin Duh, and Xavier Carreras (eds.), *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pp. 2383–2392, Austin, Texas, November 2016. Association for Computational Linguistics. doi: 10.18653/v1/D16-1264. URL <https://aclanthology.org/D16-1264/>.
- Pasko Rakic, Jean-Pierre Bourgeois, Maryellen F. Eckenhoff, Nada Zecevic, and Patricia S. Goldman-Rakic. Concurrent overproduction of synapses in diverse regions of the primate cerebral cortex. *Science*, 232(4747):232–235, 1986. doi: 10.1126/science.3952506. URL <https://www.science.org/doi/abs/10.1126/science.3952506>.
- Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H Lampert. icarl: Incremental classifier and representation learning. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pp. 2001–2010, 2017.
- Maria Refinetti, Stéphane d’Ascoli, Ruben Ohana, and Sebastian Goldt. Align, then memorise: the dynamics of learning with feedback alignment. In *International Conference on Machine Learning*, pp. 8925–8935. PMLR, 2021.
- Jill Sakai. How synaptic pruning shapes neural wiring during development and, possibly, in disease. *Proceedings of the National Academy of Sciences*, 117(28):16096–16099, 2020. doi: 10.1073/pnas.2010281117. URL <https://www.pnas.org/doi/abs/10.1073/pnas.2010281117>.
- Victor Sanh, Thomas Wolf, and Alexander Rush. Movement pruning: Adaptive sparsity by fine-tuning. *Advances in neural information processing systems*, 33:20378–20389, 2020.
- Pedro HP Savarese, Leonardo O Mazza, and Daniel R Figueiredo. Learning identity mappings with residual gates. *arXiv preprint arXiv:1611.01260*, 2016.
- Tal Schuster, Adam Fisch, Jai Gupta, Mostafa Dehghani, Dara Bahri, Vinh Q. Tran, Yi Tay, and Donald Metzler. Confident adaptive language modeling, 2022. URL <https://arxiv.org/abs/2207.07061>.
- Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In David Yarowsky, Timothy Baldwin, Anna Korhonen, Karen Livescu, and Steven Bethard (eds.), *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pp. 1631–1642, Seattle, Washington, USA, October 2013. Association for Computational Linguistics. URL <https://aclanthology.org/D13-1170/>.
- Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958, 2014. URL <http://jmlr.org/papers/v15/srivastava14a.html>.
- Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. Highway networks. *arXiv preprint arXiv:1505.00387*, 2015.

- 
- Ilya O Tolstikhin, Neil Houlsby, Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Thomas Unterthiner, Jessica Yung, Andreas Steiner, Daniel Keysers, Jakob Uszkoreit, et al. Mlp-mixer: An all-mlp architecture for vision. *Advances in neural information processing systems*, 34:24261–24272, 2021.
- Gido M Van de Ven and Andreas S Tolias. Three scenarios for continual learning. *arXiv preprint arXiv:1904.07734*, 2019.
- A Vaswani. Attention is all you need. *Advances in Neural Information Processing Systems*, 2017.
- Andreas Veit, Michael J Wilber, and Serge Belongie. Residual networks behave like ensembles of relatively shallow networks. *Advances in neural information processing systems*, 29, 2016.
- Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*, 2018a.
- Liyuan Wang, Xingxing Zhang, Hang Su, and Jun Zhu. A comprehensive survey of continual learning: Theory, method and application. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024.
- Xin Wang, Fisher Yu, Zi-Yi Dou, Trevor Darrell, and Joseph E. Gonzalez. Skipnet: Learning dynamic routing in convolutional networks. In *The European Conference on Computer Vision (ECCV)*, September 2018b.
- Sam Wass. Distortions and disconnections: Disrupted brain connectivity in autism. *Brain and Cognition*, 75(1):18–28, 2011. ISSN 0278-2626. doi: <https://doi.org/10.1016/j.bandc.2010.10.005>. URL <https://www.sciencedirect.com/science/article/pii/S0278262610001399>.
- Zuxuan Wu, Tushar Nagarajan, Abhishek Kumar, Steven Rennie, Larry S Davis, Kristen Grauman, and Rogerio Feris. Blockdrop: Dynamic inference paths in residual networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.
- Zonghan Yang, Yang Liu, Chenglong Bao, and Zuoqiang Shi. Interpolation between residual and non-residual networks. In *International Conference on Machine Learning*, pp. 10736–10745. PMLR, 2020.
- Fisher Yu, Dequan Wang, Evan Shelhamer, and Trevor Darrell. Deep layer aggregation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2403–2412, 2018.
- Alireza Zaeemzadeh, Nazanin Rahnavard, and Mubarak Shah. Norm-preservation: Why residual networks can become extremely deep? *IEEE transactions on pattern analysis and machine intelligence*, 43(11): 3980–3990, 2020.
- Sergey Zagoruyko and Nikos Komodakis. Diracnets: Training very deep neural networks without skip-connections. *arXiv preprint arXiv:1706.00388*, 2017.
- Amir R Zamir, Te-Lin Wu, Lin Sun, William B Shen, Bertram E Shi, Jitendra Malik, and Silvio Savarese. Feedback networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1308–1317, 2017.
- Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual learning through synaptic intelligence. In *International conference on machine learning*, pp. 3987–3995. PMLR, 2017.
- Xiao Zhang, Ruoxi Jiang, William Gao, Rebecca Willett, and Michael Maire. Residual connections harm generative representation learning. *arXiv preprint arXiv:2404.10947*, 2024.
- Defa Zhu, Hongzhi Huang, Zihao Huang, Yutao Zeng, Yunyao Mao, Banggu Wu, Qiyang Min, and Xun Zhou. Hyper-connections. In *The Thirteenth International Conference on Learning Representations*, 2025. URL <https://openreview.net/forum?id=9FqARW7dwB>.
- Michael Zhu and Suyog Gupta. To prune, or not to prune: exploring the efficacy of pruning for model compression. *arXiv preprint arXiv:1710.01878*, 2017.

Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *Proceedings of the IEEE international conference on computer vision*, pp. 19–27, 2015.

## A Experimental Details

**CIFAR-10 - MLP Mixer:** The MLP Mixers have 16 layers with a hidden size of 128. The patch size is 4 (the input is 32x32, 3 channels). The MLP dimension  $D_C$  is 512, while  $D_S$  is 64. We are using the AdamW optimizer Loshchilov (2017) with a maximum learning rate of 0.001 and a Cosine Scheduler with Warmup. The batch size is 64.

**BERT post-training pruning:** For Magnitude pruning, we consider the setting where the pruning happens after fine-tuning on the downstream task. For Movement pruning, we follow a gradual fine-tune and prune curriculum, where in setting (I): 20% of the parameters are pruned after each epoch, whereas in setting (II): we prune 40% of the parameters after an epoch.

**Continual Learning Experiments:** We are using the same MLP-Mixer setup with the C-far-10 experiment (see above). We train for 10 epochs in each task, using AdamW with learning rate of 0.001 and a batch size of 64. For Synaptic Intelligence we use a coefficient  $\lambda = 1$ .

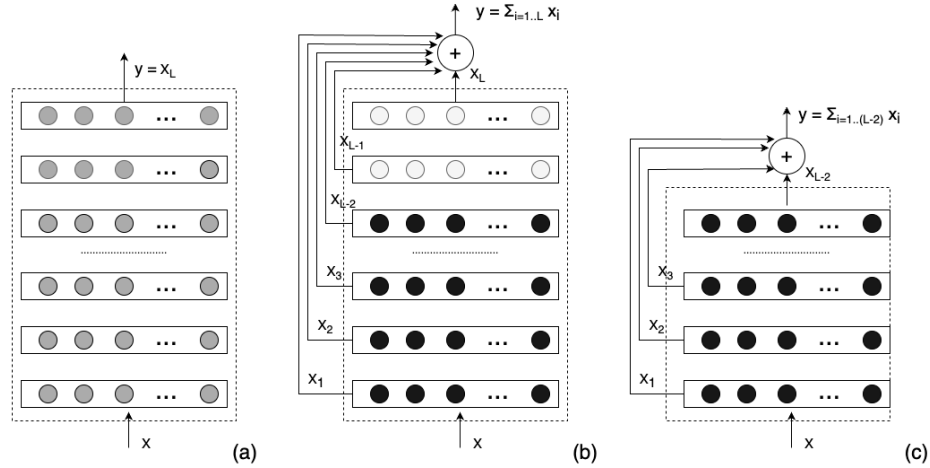


Figure 11: Main concept: (a) Start from a neural network with  $L$  layers either randomly initialized or pretrained. (b) Add residual long connections from each layer to the output of the network and sum them (also remove any existing short residual connections - if any). During training of the resulting ACN network the majority of the information (shown here as darker vs lighter circles) will naturally concentrate at the lower layers. (c) You may now safely remove the top (two in our example) layers during inference without any performance loss.

## B Gradient Propagation equations derivation

FFN forward pass

$$y_F(= x_L) = \prod_{i=1}^L w_i x_0 \quad (5)$$

FFN backward pass for weight  $i$

$$\frac{\partial y_F}{\partial w_i} = \frac{\partial y_F}{\partial x_i} \frac{\partial x_i}{\partial w_i} \quad (6)$$

---


$$\frac{\partial y_F}{\partial w_i} = \underbrace{\left( \prod_{k=i+1}^L w_k \right)}_{\text{backward term}} \underbrace{\left( \prod_{m=1}^{i-1} w_m \right)}_{\text{forward term}} x_0 \quad (7)$$

ResNet forward pass

$$x_i = w_i x_{i-1} + x_{i-1} = (1 + w_i) x_{i-1} \quad (8)$$

$$y_R = \prod_{i=1}^L (1 + w_i) x_0 \quad (9)$$

ResNet backward pass for weight  $i$

$$\frac{\partial y_R}{\partial w_i} = \frac{\partial y_R}{\partial x_i} \frac{\partial x_i}{\partial w_i} = \left( \prod_{k=i+1}^L (1 + w_k) \right) \left( \prod_{m=1}^{i-1} (1 + w_m) \right) x_0 \quad (10)$$

$$\frac{\partial y_R}{\partial w_i} = \underbrace{\left( 1 + \sum_{k=i+1}^L w_k + \sum_{i+1 \leq k < j \leq L} w_k w_j + \cdots + \prod_{k=i+1}^L w_k \right)}_{\text{backward term}} \underbrace{\left( \prod_{m=1}^{i-1} (1 + w_m) \right)}_{\text{forward term}} x_0 \quad (11)$$

or equivalently:

$$\frac{\partial y_R}{\partial w_i} = \underbrace{\left( 1 + \sum_{k=1}^{L-i+1} \text{sum of } \binom{L-i+1}{k} w \text{ k-tuples} \right)}_{\text{backward term}} \underbrace{\left( \prod_{m=1}^{i-1} (1 + w_m) \right)}_{\text{forward term}} x_0 \quad (12)$$

ACN forward pass

$$y_A = x_0 + \sum_{i=1}^L x_i = x_0 + \sum_{i=1}^L \prod_{j=1}^i w_j x_0 \quad (13)$$

ACN backward pass for weight  $i$

$$\frac{\partial y_A}{\partial w_i} = \frac{\partial y_A}{\partial x_i} \frac{\partial x_i}{\partial w_i} = \left( 1 + \sum_{k=i+1}^L \frac{\partial x_k}{\partial x_i} \right) x_{i-1} \quad (14)$$

$$\frac{\partial y_A}{\partial w_i} = \underbrace{\left( 1 + \sum_{j=i+1}^L \prod_{k=i+1}^j w_k \right)}_{\text{backward term}} \underbrace{\left( \prod_{m=1}^{i-1} w_m \right)}_{\text{forward term}} x_0 \quad (15)$$