

---

# Building and training modern ConvNet architectures from scratch

---

**Yuxin Jin**

yuxinjin@kth.se

**Yanling Lu**

yanllu@kth.se

**Madeleine Marangoz**

marangoz@kth.se

## Abstract

In this project, we developed and trained a modern convolutional neural network (ConvNet) from scratch to perform image classification on the CIFAR-10 and CIFAR-100 datasets. Starting from a basic VGG-style architecture, we systematically introduced enhancements including regularization methods (dropout, L2 weight decay, data augmentation), architectural improvements (global average pooling, strided convolutions), and advanced training strategies like cosine annealing with warm-up and restart. To further boost performance, we implemented ResNet-18 and its attention-augmented variant, SE-ResNet-18. The results highlight the benefits of deeper networks and attention mechanisms in handling complex image classification tasks.

# 1 Introduction

Convolutional Neural Networks (ConvNets) have become a method particularly for image classification tasks. In this project, we address the problem of building and training a modern ConvNet architecture from scratch to classify images in the CIFAR-10 and CIFAR-100 datasets [Krizhevsky (2009)]. The study began with a basic VGG-style convolutional network and progressively incorporated various improvements such as advanced regularization techniques (dropout, L2 weight decay, data augmentation), architectural changes (global average pooling, strided convolutions), and learning rate scheduling strategies (cosine annealing with warm-up and restart). This was further extended by including ResNet-18 and SE-ResNet-18, and evaluated their performance on both CIFAR-10 and CIFAR-100 datasets.

The results of these experiments show that combining multiple regularization strategies significantly improved test accuracy, where the best VGG-based model achieving a test accuracy of 88.67% on CIFAR-10. The ResNet-18 architecture further enhanced performance to 91.94%, and integrating SE (Squeeze-and-Excitation) blocks into ResNet yielded a peak accuracy of 92.39% on CIFAR-10. When experimenting on CIFAR-100, the SE-ResNet model achieved a test accuracy of 68.87%.

## 2 Methods

### 2.1 Dataset description

For our project, we used the CIFAR-10 dataset, which contains 60,000 32×32 color images across 10 different classes (such as airplane, car, bird, etc.), with 50,000 images designated for training and 10,000 for testing. This was mainly used for the VGG layering, meanwhile, the cifar-100 dataset was also used for the ResNet and SE layer. The cifar-100 dataset contains 100 classes containing 600 images each.

### 2.2 Data processing and Augmentation

To prepare the data for training, we first transform the loaded images and labels from the dataset to the desired data type. Data augmentation will be performed at needs to test its affect on the model performance. Each image will then be normalized using the mean and standard deviation of the dataset. Batch normalization was applied after every convolutional layer and the first fully connected layer to stabilize training.

### 2.3 Model architectures

#### 2.3.1 VGG network

The project began with a VGG-style convolutional neural network which was progressively extended with firstly testing a 1-layer VGG block that contained two 3 x 3 layers + max pooling. Later on, this was extended to a 3-layer VGG where it had double the number of filters applied after a down-sampling while in the final block a final max-pooling layer had not been included. This models were retrained using SGD optimizer with different regularization methods.

#### 2.3.2 VGG modification

One of the modification experimented with the VGG-3 was performing the down-sampling by applying the second convolution in each block with stride 2 and removing the max-pooling layer.

Another model variation tested was replacing the fully-connected layer after the last convolution layer with a global average pooling (GAP) layer. The GAP layer drastically reduces parameters so that the amount of regularization needed decreases and a higher initial learning rate would generally be preferred.

#### 2.3.3 ResNet

Residual Networks (ResNets) were proposed by He et al [He et al. (2015)]. The purpose of ResNet is to handle the problem when training a very deep convolutional network, where the performance decreases. This is often caused by the vanishing gradient problem, where gradients become too small for the network to learn effectively during backpropagation.

The motivation of ResNet is a hypothesis that convolutional networks can learn a residual mapping better than an identity mapping. To be more specific, rather than learning a complete transformation  $H(x)$ , each residual block is designed to learn a residual function  $F(x) = H(x) - x$ , leading to a reformulated output as  $H(x) = F(x) + x$ .

This structure introduces a shortcut cut, and we call it a residual connection. This is the key innovation of ResNet, and it significantly improves training stability and makes it possible to build very deep networks. Because of this design, ResNet can have hundreds of layers and still achieve high performance. At the same time, residual connections can be applied to every other Neural Network and enhance the performance in an easy way.

In our experiments, we implemented a ResNet-18 to further improve the performance.

#### 2.3.4 SENet

SENet (Squeeze-and-Excitation Networks) [Hu et al. (2017)] is a small attention block that can be added to existing convolutional neural networks to improve performance. It is a channel-wise attention mechanism and works by helping the network understand which channels (feature maps) are more important for the task.

The main idea is to give more weight to useful features and reduce the weight of less useful ones. SENet does this by implementing squeeze and excitation. Firstly squeeze the feature by using a global average pooling to get a global representation of each channel. Then, in the excitation part, learning weights are passed through two fully connected layers. These weights are used to adjust the original features by scaling.

By focusing on channel importance, SENet helps the network learn better with little extra cost. It can be easily added to most CNNs, including ResNet. In ResNet+SE 1, the SE block is added after the residual block and before the final addition with the input. Instead of adding the output directly to the input, it first goes through the SE block. The SE block reweights the feature by channel-wise attention and improves the quality of the representation.

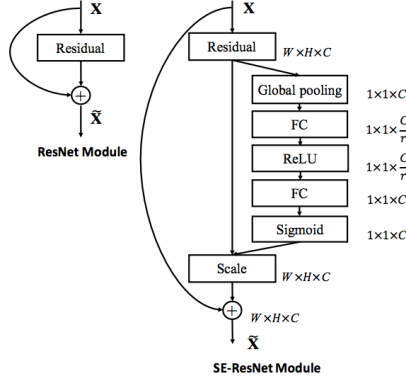


Figure 1: ResNet + SE block.

## 2.4 Learning rate scheduling

Several learning rate schedulers were studied to optimize convergence. Including cosine annealing, warm-up and restart.

## 3 Experiments

### 3.1 Implementation Details

All models were trained on Apple Silicon M2 GPU using PyTorch framework. We use gradient descent (SGD) with a momentum of 0.9. We fix the batch size to 100 and the our loss function is cross-entropy loss with L2 regularization. Up to the tests for vanilla VGG-1 and each of the basic regularization variant for VGG-3, 30 epochs were ran per test. From the improved VGG-3 model, the rest of the experiments were done using 50 epochs.

For data augmentation. During training, we apply data augmentation with random horizontal flips (50% probability), random translations (up to 4 pixels), and optional cutout (50% chance of randomly masking a 16 x 16 square region), followed by normalization; label smoothing is used when specified, and training samples are shuffled at each epoch. For validation and testing, no augmentation is applied—only normalization is performed, and the data is not shuffled.

We implement both standard ResNet-18 and SE-ResNet-18 from scratch, following the original [2, 2, 2, 2] block configuration. Each model has four convolutional stages with feature dimensions: 64, 128, 256, and 512.

### 3.2 Data

In our project, we use the CIFAR-10 dataset for training, validation, and testing. CIFAR-10 contains 60,000 32×32 color images classified into 10 classes, with 6,000 images per class. It is a widely used benchmark for image classification tasks due to its balanced class distribution and manageable size. For the training and validation data, we use all five batches of CIFAR-10 and keep the last 1,000 images as validation data. To be more specific, we use 49,000 images for training and 1,000 for validation. For the testing data, we use the test\_batch, which contains 10,000 images. We preprocess all images by dividing them by 255 to scale the values between 0 and 1, and then apply normalization to achieve a mean of 0 and a standard deviation of 1.

For extended evaluation (A part), we also conduct experiments with the CIFAR-100 dataset, which presents a more challenging classification problem. CIFAR-100 contains 60,000 32×32 color images. Unlike CIFAR-10, CIFAR-100 has 100 classes, each containing only 600 images, making the task more complex due to the smaller number of samples per class and higher inter-class similarity. For CIFAR-100, we use 49,000 images from cifar-100-python/train for training, the last 1,000 images from the same file for validation, and 10,000 images from cifar-100-python/test for testing. We apply the same preprocessing to CIFAR-100. Using both datasets allows us to evaluate the generalization and robustness of our model under different levels of difficulty.

## 4 Results

We conducted a set of experiments to evaluate the performance of various convolutional neural network configurations on the CIFAR-10 dataset. The models were improved by introducing different regularization techniques using SGD optimizer.

### 4.1 Results of the model with 1 layer of VGG

The vanilla model is the model using only one layer of VGG and it exhibited signs of underfitting, with a relatively good accuracy and relatively high loss. The plots for this configuration is given in Fig. 2. The vanilla model had a test accuracy of 65.62%, a test loss of 1.0403 and a test cost of 1.1886.

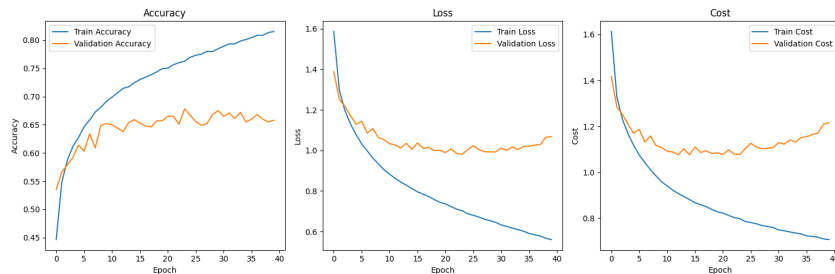


Figure 2: Training and validation curves for the Vanilla Model using only one layer of VGG.

### 4.2 Results of the model with 3 layers of VGG

#### 4.2.1 VGG-3 with basic regularization variants

After testing the vanilla model, more layers of VGG were added where in total there was 3 VGG layers. Table 1 presents the test accuracy, loss, and cost for each model variant where all configurations contained a 3-layer VGG:

Model Variant (3-layer VGG)	Test Accuracy	Test Loss	Test Cost
L2 Regularization Only	0.7840	0.7726	2.2058
Dropout Only	0.8500	0.4603	0.4603
L2 + Dropout	0.8527	0.4790	1.9954
Augmentation Only	0.8654	0.4166	0.4166
<b>Improved Model (All Combined)</b>	<b>0.8845</b>	<b>0.3521</b>	<b>2.4688</b>

Table 1: Comparison of test metrics for each model variant

Testing the model with 3 layers of VGG using L2 regularization only improved accuracy but led to overfitting as seen from the increasing validation cost. Dropout alone resulted in well-balanced training and validation curves, achieving high accuracy with the lowest cost. When only applying data augmentation, it also significantly increased generalization which gave the second-best performance across all metrics.

The improved model, which combined multiple strategies including batch normalization, dropout, L2 regularization, and data augmentation, achieved the highest accuracy (0.8845). However, its cost was disproportionately high (2.4688). Below are the plots for this configuration.

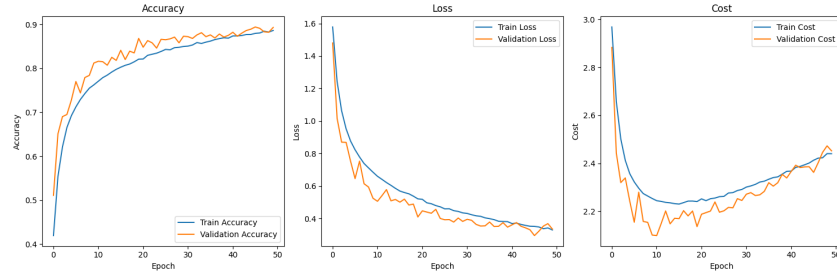


Figure 3: Training and validation curves using three layers of VGG combining L2 regularization, dropout and data augmentation.

#### 4.2.2 VGG-3 with more advanced regularization/model variants

A few advanced techniques were tested on top of the VGG-3 model with the basic regularizations.

Two types of data augmentation - cutout and label smoothing were added to the basic regularization presented in the previous section. With the same parameter settings ( $lr = 0.001$ ,  $lam = 0.001$ ), the corresponding test results are shown in Table 2. Comparing with the 88.45% test accuracy achieved with only the basic regularizations, none of the addition augmentations improved the performance.

Data Augmentation Variants	Test Accuracy	Test Loss	Test Cost
Basic & Cutout	0.8718	0.3908	2.4800
Basic & Label smoothing (0.1)	0.8838	0.3553	2.4719

Table 2: Comparison of test metrics for adding more data-augmentation

For the learning rate scheduling, cosine annealing with warm-up and restart were added to the basic regularization under the same settings. With  $lr_{max} = 0.001$  and  $lr_{max} = 0.00001$ , different restart schedules were tested as shown in Table 3. However, none of the schedule settings could improve the performance either.

LR Scheduling Variant	Test Accuracy	Test Loss	Test Cost
$T_0 = 10, T_{mult} = 2$	0.8816	0.3676	2.2330
$T_0 = 15, T_{mult} = 2$	0.8636	0.4039	2.2130
$T_0 = 20, T_{mult} = 1$	0.8759	0.3683	2.1892

Table 3: Comparison of test metrics for different schedule parameters

The third type of changes applied to the basic VGG-3 model is the down-sampling via stride 2 on the second convolution in each VGG block. This model was tested using the basic regularization and achieved a test accuracy of 88.67%, a loss of 0.3683 and cost of 2.1892 - despite having greater loss, this model results in higher accuracy than the basic VGG-3 model.

The last changes tested was replacing the FC layer after the last convolution layer with a GAP layer. This change was applied on top of the strided convolution for down-sampling. A few tests were

conducted for different dropout rate and L2-regularization combinations using  $lr = 0.005$ . The overall performance of this model is, however, worse than that of the basic model.

Dropout & Lam Variant	Test Accuracy	Test Loss	Test Cost
Basic (no dropout, $lam = 0.001$ )	0.8673	0.4128	2.4181
dropout rate= 0.1, $lam = 0.001$	0.8662	0.4489	2.5287
dropout rate= 0.1, $lam = 0.0005$	0.8838	0.3864	1.4241
dropout rate= 0.2, $lam = 0.0005$	0.8769	0.3942	1.4200
dropout rate= 0.2, $lam = 0.001$	0.8741	0.4109	2.4635

Table 4: Comparison of test metrics for different regularization tuning

### 4.3 Results of the vanilla ResNet model

The ResNet-18 architecture was evaluated on CIFAR-10 and CIFAR-100 using tuned settings: L2 regularization ( $\lambda = 0.001$ ), label smoothing, cosine annealing from 0.1 to 0.001, and a learning rate restart at epoch 10. As shown in Table 5. The model achieves a test accuracy of 91.94% on CIFAR-10 and 69.88% on CIFAR-100. This shows that the architecture can effectively be applied to image classification tasks and achieves quite promising results. The lower test accuracy on CIFAR-100 is due to the complexity of the CIFAR-100 dataset, and limited amounts of samples in each class. The number is also reasonable comparing with the results from other models [PapersWithCode (2025)]. Fig. 4 shows the accuracy and loss plot of the model on CIFAR-10 (on the left) and CIFAR-100 (on the right).

Model	Dataset	Test Accuracy	Test Loss
ResNet-18	CIFAR-10	0.9194	0.4720
	CIFAR-100	0.6988	1.8710
ResNet-18 + SE	CIFAR-10	0.9239	0.4341
	CIFAR-100	0.6887	1.8773

Table 5: Comparison of ResNet18 on different datasets

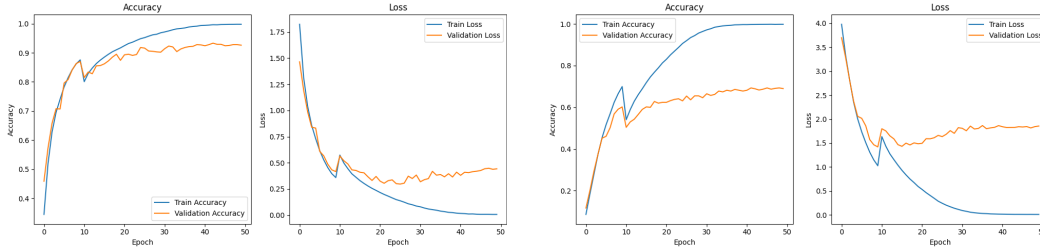


Figure 4: Training and validation curves for ResNet-18 on CIFAR-10 and -100

### 4.4 Results of the SENet Model

We evaluate ResNet-18 with SE on CIFAR-10 and CIFAR-100 with the same settings as ResNet-18. As shown in Table 5, the model achieves a higher test accuracy of 92.39% on CIFAR-10, but a lower test accuracy of 68.87% on CIFAR-100. The potential reason will be discussed in the next section. Figure below shows the accuracy and loss plot of the model on CIFAR-10 (on the left) and CIFAR-100 (on the right).

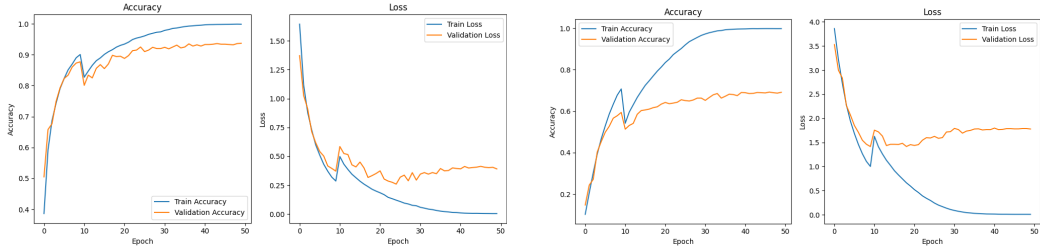


Figure 5: Training and validation curves for ResNet-18 + SE on CIFAR-10 and -100

## 5 Discussion

For the 3-block VGG architecture, the L2 Regularization helped reduce overfitting, but not sufficiently by itself. When all strategies were combined, dropout, L2, data augmentation, and batch normalization, the model reached its highest VGG-based accuracy at 88.45%. These experiments showcase how sensitive VGG networks are to regularization and architectural tweaks. Beyond a certain point however, not much improvement occurs since returns diminished or even declined, possibly due to over-regularization for short training. While the VGG-based models show a relatively high accuracy on CIFAR-10, this also helped motivate to further transition into deeper models such as ResNet and SE-ResNet.

Another observation is that despite replacing max-pooling with a strided convolution for down sampling further improved the test accuracy of VGG-3, replacing the FC layer with the GAP layer resulted in worse performance. This can happen especially with simple dataset. The strength of GAP lies in improving generalization and lower fitting at a slight cost to peak accuracy. The drastic reduce in parameter counts lead to both higher computational efficiency and the higher potential of lacking expressive power.

The test accuracies of ResNet and SENet was as expected with over 90% for CIFAR-10 and close to 70% for CIFAR-100. The reason why the two models perform relatively better on one dataset but worse on the other may be explained by the nature of the two datasets. While CIFAR-10 has 10 broad classes, meaning discriminative features are easier to learn, CIFAR-100 has 100 fine-grained classes, requiring more nuanced feature extraction. SE blocks usually help re-weight channels to focus on the most discriminative features, which is highly effective when class distinctions are clear. However, in CIFAR-100, the channel interdependencies are more complex that SE blocks may overfit or misweight less prominent but critical features.

To further generalize the result of ResNet-based model to CIFAR-100 or other complicated image classification datasets, we believe one approach is to replace ResNet18 with ResNet50 or 101. Since deeper architecture with residual connections has been shown to perform well according to previous studies. Training for longer times and adding more data augmentations might help as well to increase the representation of the limited data.

## 6 Conclusion

This project demonstrated the process of building and iteratively improving a ConvNet architecture for image classification. Starting with a baseline VGG model, we showed how incremental additions of regularization techniques and architectural changes could significantly improve performance. The transition to deeper models like ResNet-18, and further enhancements with SE blocks, led to substantial accuracy gains, especially on CIFAR-10. While SE-ResNet models improved performance on simpler datasets, their benefits were less clear on CIFAR-100, where overfitting and subtle class distinctions posed greater challenges. These findings suggest that future improvements could involve deeper ResNet variants (e.g., ResNet-50/101), longer training durations, or more diverse data augmentations to better generalize to complex datasets. Overall, the project validated the effectiveness of modern ConvNet design principles and the importance of systematic experimentation in achieving high performance.

## References

- Alex Krizhevsky. *Learning Multiple Layers of Features from Tiny Images*. Technical Report, University of Toronto, 2009. Available: <https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf>.
- Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. *Deep Residual Learning for Image Recognition*. CoRR, abs/1512.03385, 2015. Available: <http://arxiv.org/abs/1512.03385>.
- Jie Hu, Li Shen, and Gang Sun. *Squeeze-and-Excitation Networks*. CoRR, abs/1709.01507, 2017. Available: <http://arxiv.org/abs/1709.01507>.
- Papers With Code. *Image Classification on CIFAR-100*. 2024. Available: <https://paperswithcode.com/sota/image-classification-on-cifar-100> [Accessed: 19 May. 2025].