A Report on the CIFAR-10 dataset

Image recognition via a Convolutional Neural Network

Dominik Sieber

December 1, 2025

Assessment rubric (25 points total)

Block	Pts	What must be shown / explained to earn points
Model and hyperparameter study	**12**	*CNN architecture for images* that introduces convolution, pooling and suitable nonlinearities or normalisation, with a short justification of design choices **(2)**. *Data Split* into train / validation / test split **(1)**. *Baseline MLP on* `CIFAR10` with description of the single hidden layer network, chosen metric, learning curves and baseline accuracy. A quantitative comparison to the baseline MLP with a simple CNN **(4)**. *Ablation study of the model* where building blocks are added step by step and their impact on validation performance is shown in a table or figure and discussed so that it is clear which components matter and which do not (reasonable in regards to computational efficiency)**(3)**. *Hyperparameter search* that compares different model sizes, optimisers or settings such as learning rate, batch size or weight decay through learning curves and final accuracies, with an explanation of how these choices affect convergence speed and final performance **(2)**.
Data augmentation	**4**	*Augmentation* of the data such as horizontal flip, random crops, small translations or color jitter and their comparison performance to non augmented data (shown in results). Short explanation why this is important. **(4)**.
Training Results	**4**	*Results* of the best performing model architecture including loss trajectories, image classification examples and potentially confusion matrix both for augmented and non augmented data **(4)**.
Presentation	**5**	Clear, concise figures with informative captions. General layout and language usage

1 Convolutional neural networks for images

1.1 Discrete convolution on images

For a single image with height H and width W a discrete two dimensional convolution between an input $x \in \mathbb{R}^{H \times W}$ and a kernel (or filter) $w \in \mathbb{R}^{k_h \times k_w}$ is defined as

$$(x * w)_{i,j} = \sum_{u=0}^{k_h-1} \sum_{v=0}^{k_w-1} w_{u,v}\, x_{i+u-\lfloor k_h/2 \rfloor,\, j+v-\lfloor k_w/2 \rfloor} \tag{1}$$

for output locations (i, j) where the indices on the right hand side remain inside the image domain. In practice convolution is applied with appropriate padding which virtually extends the input, for example by zeros, so that also boundary pixels can be computed.

Color images have three input channels. In that case each convolutional filter has a channel dimension and the convolution sums over spatial and channel indices

$$y_{c^{\text{out}},i,j} = \sum_{c^{\text{in}}=1}^{C_{\text{in}}} \sum_{u,v} w_{c^{\text{out}},c^{\text{in}},u,v}\, x_{c^{\text{in}},i+u,j+v} \tag{2}$$

which produces an output tensor $y \in \mathbb{R}^{C_{\text{out}} \times H' \times W'}$ with C_{out} learned feature maps. The same kernel weights are reused at all spatial positions. This weight sharing gives CNNs a **strong inductive bias** for **translation equivariance** and reduces the number of parameters compared to fully connected layers that **ignore spatial structure.**

The stride controls by how many pixels the kernel is shifted between two positions. A stride larger than one reduces the spatial resolution and therefore the computational cost. Padding controls how the input is extended at the border and influences both the output size and how much information from the border enters the computation.

1.2 Typical CNN building blocks

Modern CNNs are built from a small set of standard components that are repeated in a structured way. Table 1 summarises the building blocks that are used throughout this report.

The main architectural degrees of freedom are the number of convolutional blocks, their channel width, kernel sizes, pooling strategy and the depth and width of the dense classifier. Hyperparameters of the optimiser and learning rate schedule further influence how well a given architecture can be trained.

2 Data set and experimental protocol

CIFAR10 consists of 60 000 color images of size 32×32 pixels with three channels and ten object classes. There are 50 000 training images and 10 000 test images. The classes are balanced in the original split.

The predefined test set is used only once for the final evaluation of the best model. For model

Block	Role and effect
Convolution $(k_h \times k_w)$	Learns spatial filters on local neighbourhoods of size $k_h \times k_w$. The number of output channels controls how many distinct features are extracted.
Nonlinearity (ReLU)	Applies an elementwise activation $\varphi(z) = \max(0, z)$ which introduces nonlinearity and improves gradient flow for positive activations.
Batch normalisation	Normalises activations per mini batch to zero mean and unit variance and then applies a learned affine transform. This stabilises optimisation and allows larger learning rates.
Pooling (max pooling)	Aggregates information in local windows, typically of size 2×2. Max pooling keeps the maximum activation which focuses on the strongest response and reduces spatial resolution.
Dropout	Randomly sets activations to zero with a fixed probability during training. This acts as a regulariser and reduces overfitting.
Global average pooling	Averages each feature map over all spatial positions which turns a tensor of shape $C \times H \times W$ into a vector in $\mathbb{R}^C$. This replaces large fully connected layers and encourages the network to learn class specific feature maps.
Fully connected classifier	Maps the final feature vector to class logits through one or more dense layers and a final linear layer with ten outputs for CIFAR10.

Table 1: Standard building blocks used in the convolutional networks in this report.

selection and ablation studies the 50 000 training images are split into a training part and a validation part. A random split of $90\,\%$ training and $10\,\%$ validation is used which yields 45 000 training images and 5000 validation images. All splits are stratified by class label.

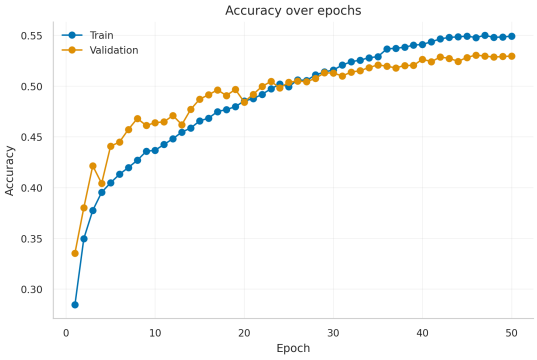
All models are trained with cross entropy loss and evaluated with classification accuracy on the validation set. For the hyperparameter search and ablation runs the number of epochs is limited to 20 for computational efficiency. The baseline and simple CNN models are trained for 50 epochs and the final selected model is trained for 200 epochs.

3 Baseline models

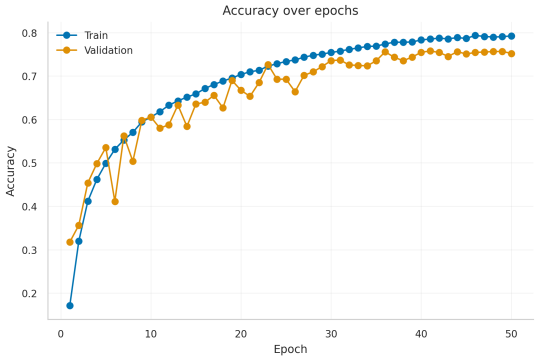
As a reference a simple multilayer perceptron (MLP) that ignores spatial structure is trained. Each input image is flattened into a vector in $\mathbb{R}^{3072}$ and passed through a single hidden layer with 512 units and ReLU activation. The configuration is

- one hidden dense layer with 512 units
- ReLU activation
- dropout with rate 0.2 on the hidden layer
- He normal weight initialisation for all linear layers
- biases initialised to zero
- output layer with ten logits and no hidden nonlinearity.

The model is trained with an adaptive optimiser and a fixed learning rate for 50 epochs. Figure 1a shows the training and validation loss and accuracy over epochs. The validation accuracy converges to roughly 52 %.



(a) Learning curves for the fully connected baseline. Training and validation loss and accuracy over 50 epochs on CIFAR10. The model reaches about 52 % validation accuracy.



(b) Learning curves for the simple CNN baseline. Training and validation loss and accuracy over 50 epochs. The model reaches around 75 % validation accuracy and clearly outperforms the fully connected baseline.

Figure 1: Comparison of learning curves for the fully connected MLP (left) and the simple CNN (right) baselines on CIFAR10. Both models are trained for 50 epochs, but the CNN benefits from convolutional structure and achieves substantially higher validation accuracy.

3.1 Simple convolutional baseline

The first convolutional model already follows common CNN design principles. It consists of three convolutional blocks followed by two dense layers and a linear classifier. The convolutional part uses

- block 1: 32 channels, 3×3 kernel, stride 1, same padding, batch normalisation, ReLU, dropout rate 0.1, 2×2 max pooling
- block 2: 64 channels, same convolutional hyperparameters, dropout rate 0.2, 2×2 max pooling
- block 3: 128 channels, same convolutional hyperparameters, dropout rate 0.3, 2×2 max pooling.

The output of the last pooling layer is flattened and passed through two dense layers with 256 and 128 units and ReLU activation.

This simple CNN is trained in the same way as the MLP above. Figure 2b shows the learning curves. The validation accuracy reaches roughly 75 %. This demonstrates that convolution, pooling and local receptive fields alone already yield a substantial improvement over the fully connected baseline even without thorough tuning.

4 Optimization of the Model

The next step is a more systematic exploration of model complexity and optimisation hyperparameters. The search space is specified in terms of convolutional blocks, dense blocks, learning rate, weight decay, optimiser and learning rate schedule. The evaluation metric is validation accuracy after 20 epochs.

4.1 Search space and procedure

The convolutional part is chosen from three families of architectures

- two block variant: two convolutional blocks with features $(32, 64)$, 3×3 kernels, max pooling and dropout rates $(0.1, 0.2)$
- three block variant: three convolutional blocks with features $(32, 64, 128)$ and dropout rates $(0.1, 0.2, 0.3)$, otherwise identical
- deep six block variant: six convolutional blocks with features $(64, 64, 128, 128, 256, 256)$. Every second block applies 2×2 max pooling and dropout, starting from the second block. Dropout rates increase from zero in the first block to 0.25 in the last block.

The dense classifier is chosen from two options

- single dense layer with 512 units, ReLU, dropout rate 0.5
- two dense layers with 512 and 256 units and dropout rates 0.4 and 0.3.

For optimisation the following hyperparameters are varied

- learning rate in $\{10^{-4}, 10^{-3}, 10^{-2}\}$
- weight decay in $\{10^{-4}, 5 \cdot 10^{-4}\}$
- optimiser in `AdamW` and `RMSprop`
- learning rate schedule in constant schedule and cosine decay.

Up to one hundred random combinations from this grid are trained for 20 epochs on the training set and evaluated on the validation set. The best validation accuracy after 20 epochs is used to rank configurations.

4.2 Search results

A hyperparemter search as well as an ablation study over the same set of parameters was done to find the best configuration of hyperparameters like the optimiser and learning rate but als the complexity of the CNN network. To compare the results, the training was limited for each combination to 20 epochs.

The main observations are

- architectures with more convolutional blocks and higher feature counts consistently outperform shallower networks at the same training budget
- AdamW with moderate weight decay and cosine learning rate decay yields better performance than RMSprop or constant learning rates for the same architecture
- too large learning rates such as 10^{-2} often lead to unstable learning or significantly worse final accuracy within 20 epochs.

Based on these results a deep architecture with six convolutional blocks, global average pooling and a single dense layer with 512 units is selected for further experiments. The corresponding optimiser settings are AdamW with learning rate $3 \cdot 10^{-4}$, weight decay $5 \cdot 10^{-4}$ and cosine decay schedule with warmup, as detailed in Section 6.

5 Data augmentation

Data augmentation aims to expand the effective training set by applying label preserving transformations to the images. For CIFAR10 many variations of the same object under small translations or rotations are plausible, and the class label is invariant under horizontal flips for most classes.

Let x denote an image and t a stochastic transformation sampled from a set of allowed transformations. During training the network does not see x directly but $\tilde{x} = t(x)$. The loss is still computed with the original label y. The training objective becomes an expectation over transformations

$$\mathcal{L}(\theta) = \mathbb{E}_{(x,y)} \, \mathbb{E}_t \big[\ell\big(f_\theta(t(x)), y\big) \big] \tag{3}$$

where f_θ denotes the network and ℓ the cross entropy loss. This encourages invariance and generalization capabilities of the network to the transformations in the augmentation set.

In this project the following augmentations are used

- random crops of size 32×32 from padded images
- random horizontal and vertical flips
- small random rotations
- cutout style random erasing where a small square region is set to zero.

The strength of each augmentation is limited so that the transformed images still resemble realistic variants of the original objects. Stronger distortions could lead to distribution shift between training and validation data.

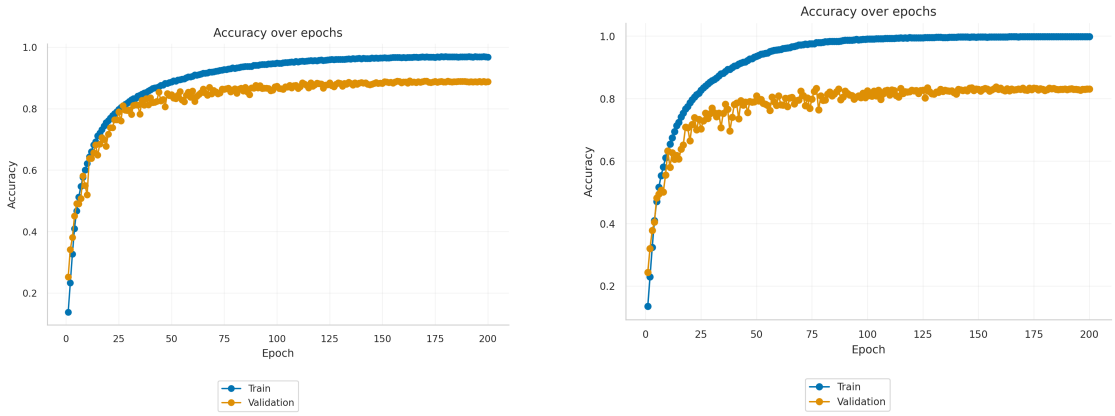
6 Final model and results

For comparison the same architecture is trained without data augmentation under otherwise identical settings. Both models are trained with early stopping deactivated in order to observe the full learning dynamics.

6.1 Learning curves and quantitative results

Figure 3 compares training and validation accuracy for the models with in Figure 3a and without in Figure 3b augmentation. Without augmentation the model reaches roughly 82 % validation accuracy and then overfits, as indicated by a continuing increase in training accuracy while validation accuracy stagnates or slightly decreases. With augmentation the model reaches about 90 % validation accuracy

and the gap between training and validation curves is reduced but still there, which nonetheless shows the regularising effect of augmentation.

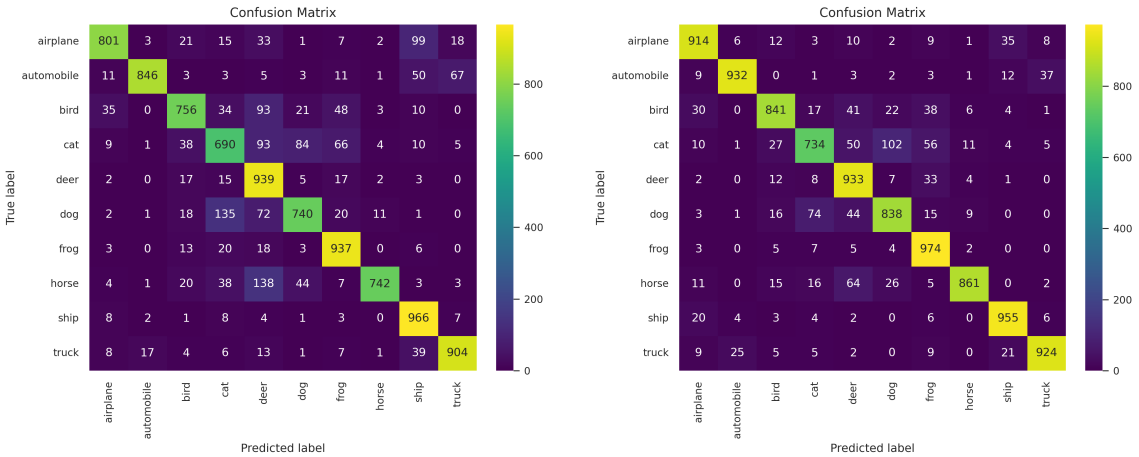


(a) (b)

Figure 2: Learning curves of the final architecture with (a) and without (b) data augmentation over 200 epochs. Data augmentation improves validation accuracy from about 82 % to about 90 % and reduces overfitting.

6.2 Qualitative analysis and confusion matrices

To better understand the remaining errors, the confusion matrices for both models are inspected. Figure 4 shows an example of the corresponding confusion matrices.



(a) Confusion without augmentation (b) Confusion with augmentation

Figure 3: Confusion matrices for the final architecture. Data augmentation reduces confusions between visually similar classes and yields a more diagonal confusion matrix.

The confusion matrices show that most errors occur between semantically and visually similar classes such as dogs and cats or trucks and automobiles. The model trained with augmentation exhibits fewer such confusions and a clearer diagonal structure, which reflects improved class separation.

6.3 Feature visualisation

Finally the internal feature maps of the augmented model are inspected for a single input image. Figure 5 illustrates how the activations evolve across the convolutional layers. The first panel shows the input image and one early feature map. The remaining panels show representative channels from deeper layers.

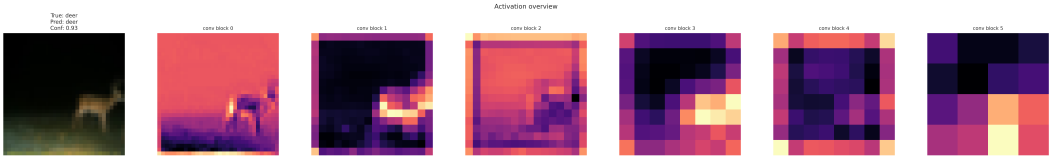


Figure 4: Intermediate activations of the augmented model for a single input image example based on a deer. Early layers respond mainly to local edges and color contrasts. Deeper layers exhibit more structured patterns that highlight object parts and shapes. Global average pooling aggregates these spatial patterns into class specific features.

The visualisation confirms the expected behaviour of a CNN. Early layers extract generic low level features such as edges and corners, while deeper layers form more abstract and object specific representations. These qualitative observations support the quantitative improvements obtained through deeper architectures and augmentation.

7 Conclusion

Starting from a simple fully connected baseline this report demonstrates how convolutional architectures, careful hyperparameter selection and data augmentation improve image classification performance on CIFAR10. Convolution and pooling alone already provide a strong gain over a dense network that ignores spatial structure. A systematic hyperparameter search and ablation study shows that sufficient depth, appropriate regularisation and a cosine decay learning rate schedule with AdamW are important for reliable optimisation.

Data augmentation further increases validation accuracy and reduces overfitting by enforcing invariance under realistic transformations. Qualitative analysis through confusion matrices and feature visualisation confirms that the final model learns meaningful hierarchical representations of the images.

References

[1] Lecture notes from Markus Schmitt for Modern Machine Learning, 2025.