ChatGPT

# Improving CIFAR-10 Training in `cifar-susume`

The **cifar-susume** project's current training setup is not reaching the 90–91% CIFAR-10 test accuracy achieved by well-tuned models. By comparing `model.py`, `data.py`, and the **training configs** to high-accuracy implementations, we can pinpoint gaps. High-performing CIFAR-10 classifiers (e.g. ResNets) use specific architectural choices, aggressive data augmentation, and optimized hyperparameters. Below, we outline the likely deficiencies and **recommend concrete changes** to reliably push test accuracy above 90%.

## Model Architecture and Network Depth

- **Use a proven deep architecture (ResNet or similar):** The repository's Model 6 is a ResNet-20, but it must precisely mirror the **original ResNet design** for CIFAR-10. A properly configured 20-layer ResNet can achieve ~8.75% test error (~91.3% accuracy) on CIFAR-10 [1]. Ensure each residual block has **batch normalization** and **ReLU** in the correct order (as in Keras/TensorFlow ResNet implementations) and use **1×1 projection shortcuts** when downsampling so that skip connections match feature map dimensions [2]. In one experiment, adopting Keras's official ResNet code yielded an **8% accuracy boost** over a custom CNN [3] – underscoring the importance of correct residual architecture.

- **Increase network depth or width if possible:** If 20 layers is not consistently hitting 90%, consider a deeper variant like **ResNet-32 or ResNet-44** (which reach ~92%+ accuracy) [1]. Similarly, ensure you're not inadvertently limiting capacity – for example, if any model variant uses **depthwise separable convolutions** or reduced channels (as hinted by "m4/m5" models), recognize that these lightweight layers trade accuracy for efficiency. For maximum accuracy, standard convolutional layers and a sufficiently large model (e.g. Wide-ResNet or deeper ResNet) are preferable [4]. In short, do not under-size the model – CIFAR-10 networks tend to require **>=1e6 parameters and 20+ layers** to cross the 90% threshold [5].

- **Avoid excessive dropout in convolutional layers:** If the current model uses dropout (`v0_drop` config), consider removing or reducing it. The original ResNet paper achieved best results **without any dropout** on CIFAR-10 [6], relying on batch norm and data augmentation for regularization. Unneeded dropout can slow training or hurt accuracy. Instead, focus on other regularization techniques (data augmentation, weight decay) which are more effective for CNNs [2].

## Data Preprocessing & Augmentation

- **Normalize inputs appropriately:** Ensure the data pipeline converts images to the standard scale (0–1 or -1 to 1) and ideally performs per-channel mean subtraction. While this input normalization has less impact when using batch normalization (one study found no significant difference due to BN's effect [7]), it's good practice for stable training. In Keras, avoid misusing `featurewise_center/std` without fitting the stats – if those were enabled and not computed, it could hinder learning [8].

A safe approach is to manually normalize using the known CIFAR-10 mean/std or let batch norm handle it implicitly.

- **Adopt** standard data augmentation **for CIFAR-10:** Insufficient augmentation can lead to overfitting (100% train accuracy but low test accuracy) [9] [10]. At minimum, apply the **classic augmentation used by ResNet**: pad each image by 4 pixels on each side (or resize to 36×36), then take a random 32×32 crop, and randomly **horizontal flip** each image [11] [12]. This basic augmentation scheme was used to train ResNets to ~6–7% error rates on CIFAR-10 [5] [11]. For example, using horizontal flips and small random shifts (cropping) improved a model's accuracy substantially in practice [8]. Make sure your `data.py` pipeline includes these transformations (e.g. via TensorFlow's `ImageDataGenerator` or PyTorch `transforms.RandomCrop/RandomHorizontalFlip`). This will increase dataset variety and reduce overfitting.

- **Leverage advanced augmentation techniques:** Once basic augments are in place, you can further boost generalization with modern techniques. Consider adding **Cutout** (randomly masking out a 8×8 or 16×16 patch of the image during training) or **MixUp** augmentation. These have been shown to improve CIFAR-10 accuracy by making the model more robust to occlusions and forcing it to learn combinational features [13]. Similarly, moderate **random rotations** or color jitter can be tried. The StackOverflow guidance specifically suggests heavy augmentation (rotation, cutout, mixup) for boosting generalization [13]. Be careful to introduce one technique at a time and evaluate – each can give a few percentage points of gain when tuned properly.

## Training Configuration & Hyperparameters

- **Use the right optimizer and learning rate schedule:** A major reason for sub-90% accuracy can be an inadequate learning rate policy. It's recommended to train with **SGD + momentum (0.9)** and an aggressive **learning-rate schedule**, as done in the ResNet paper [6]. Start with a high learning rate (e.g. 0.1) and **reduce it by a factor of 10 at least twice during training**. For example, the original CIFAR-10 training ran ~164 epochs, dropping LR at 32k and 48k iterations (roughly at 50% and 75% of training) [2]. You can mimic this by decaying at ~epoch 100 and 150 in a 200-epoch run, or use a **cosine annealing** schedule for gradual decay. *Do not keep a constant learning rate* — that often leaves the model short of its best accuracy. In fact, reproductions have found that using **SGD with scheduled LR drops yields >3% higher accuracy** than Adam with a fixed LR when training CIFAR-10 ResNets [14] [15]. If you currently rely on Adam, either switch to SGD or implement a proper LR schedule (e.g. **One-Cycle policy** or step decay). Fast.ai's "super-convergence" approach (one-cycle with a very large LR up front) is an option [16], but the simplest fix is to start with the known good schedule from literature (e.g. 0.1 -> 0.01 -> 0.001 at the prescribed epochs [2]).

- **Train for enough epochs (don't stop too early):** Hitting 90%+ may require on the order of **200 epochs** of training with the above LR schedule. If your current runs are much shorter (e.g. 50 epochs), the model likely hasn't had the chance to converge at a low learning rate. High-accuracy implementations typically train to the point where training loss is near zero [17]. For instance, one guide follows 0.1 LR until ~90 epochs, 0.01 until ~135, and 0.001 until 150+, by which point the network reaches minimal training loss and a low error plateau [15]. Ensure your config's `epochs` setting is sufficient and that you're not terminating training while the test accuracy is still climbing. With a proper schedule, **more epochs (with lower LR) will squeeze out the last bits of accuracy**.

- **Apply weight decay (L2 regularization):** Check that weight decay is enabled in your training config – this is critical for CIFAR-10. The ResNet authors used **weight decay = 1e-4** on all weights [6], which helps prevent overfitting. In Keras, this means adding an `l2(1e-4)` kernel regularizer to Conv and Dense layers (or in optimizer if using AdamW). If your config's "v0_l2" variant wasn't active, make it the default. Proper weight decay combined with batch norm let the network fit the training data fully without overfitting the test set [9] [10]. (Note: ensure you do **not** apply L2 regularization to batch norm parameters or biases in Keras, only to weights – the Keras ResNet reference excludes BN gamma/beta from decay). Using weight decay will likely improve your test accuracy by a few points by reducing the gap between train and test performance.

- **Batch size considerations:** CIFAR-10 models typically use a batch size around 128 images [2]. If you are using a very small batch (e.g. 32), training will be noisier and might require more epochs to reach the same accuracy. Conversely, extremely large batches (>>128) can require adjusting the learning rate schedule (or use gradient accumulation to simulate smaller batches). Sticking to the **128–256 range** is a good balance. This was the batch size used to achieve state-of-the-art results in the literature [6]. If you do change batch size, scale the initial learning rate approximately linearly with batch size (for example, if doubling batch to 256, you might try LR ~0.2) to maintain effective learning rate per sample.

- **Leverage ensemble-style techniques after fixing training** (optional): Once the above changes are made, your single-model test accuracy should be ~90% or higher **before any test-time tricks**. At that point, you can consider techniques like **Stochastic Weight Averaging (SWA)** and **Test-Time Augmentation (TTA)** to push a bit further. SWA can be used near the end of training to average multiple model checkpoints into a flatter minima – ensure you conduct SWA after the learning rate has been lowered significantly (otherwise SWA won't help much). TTA (e.g. averaging the predictions of an image and its flipped/multicropped versions) can also provide a minor boost (~0.5–1%). However, these methods **cannot compensate for missing fundamentals**. They are best applied once the model, data aug, and schedule are optimal. In summary, focus on the core training improvements first; with those in place, SWA and TTA become the icing on the cake (useful for squeezing out an extra ~1% accuracy in the final evaluation [13]).

By implementing the above changes – **a stronger ResNet-based model, proper data normalization & augmentation, a tuned learning rate schedule, momentum SGD with weight decay, and sufficient training duration** – the model should close the performance gap. These practices align with what the original ResNet CIFAR-10 training and other 90%+ accuracy solutions employed [2] [15]. Adopting them will make it highly likely to hit or exceed **90–91%** test accuracy on CIFAR-10, moving the cifar-susume project into the expected performance range. Each recommendation is actionable in the code/config (e.g. change optimizer settings, add augmentation in `data.py`, modify training schedule in the config), so the path to higher accuracy is clear. With a bit of hyperparameter tuning around these baselines, you should reliably achieve the target accuracy milestone.

**References:** Key sources compared include the original ResNet paper [18] [1], expert advice on training ResNets [10] [13], and community reproductions of CIFAR-10 benchmarks [14] [15], all of which consistently emphasize the points above. By aligning the `cifar-susume` training pipeline with these best practices, you can confidently boost CIFAR-10 performance into the 90%+ regime.

---

1 2 5 6 11 18 arxiv.org
https://arxiv.org/pdf/1512.03385

3 8 GitHub - hideyukiinada/cifar10: Test various techniques to assess impact on CIFAR10 accuracy using Keras. Accuracy is 91%+ for top scripts.
https://github.com/hideyukiinada/cifar10

4 7 12 14 15 17 Lessons learned from reproducing ResNet and DenseNet on CIFAR-10 dataset | by Arcanum AI | Medium
https://arcanumai.medium.com/lessons-learned-from-reproducing-resnet-and-densenet-on-cifar-10-dataset-44250211264b

9 10 13 16 deep learning - Why the resnet110 I train on CIFAR10 dataset only get 77% test acc - Stack Overflow
https://stackoverflow.com/questions/58986583/why-the-resnet110-i-train-on-cifar10-dataset-only-get-77-test-acc