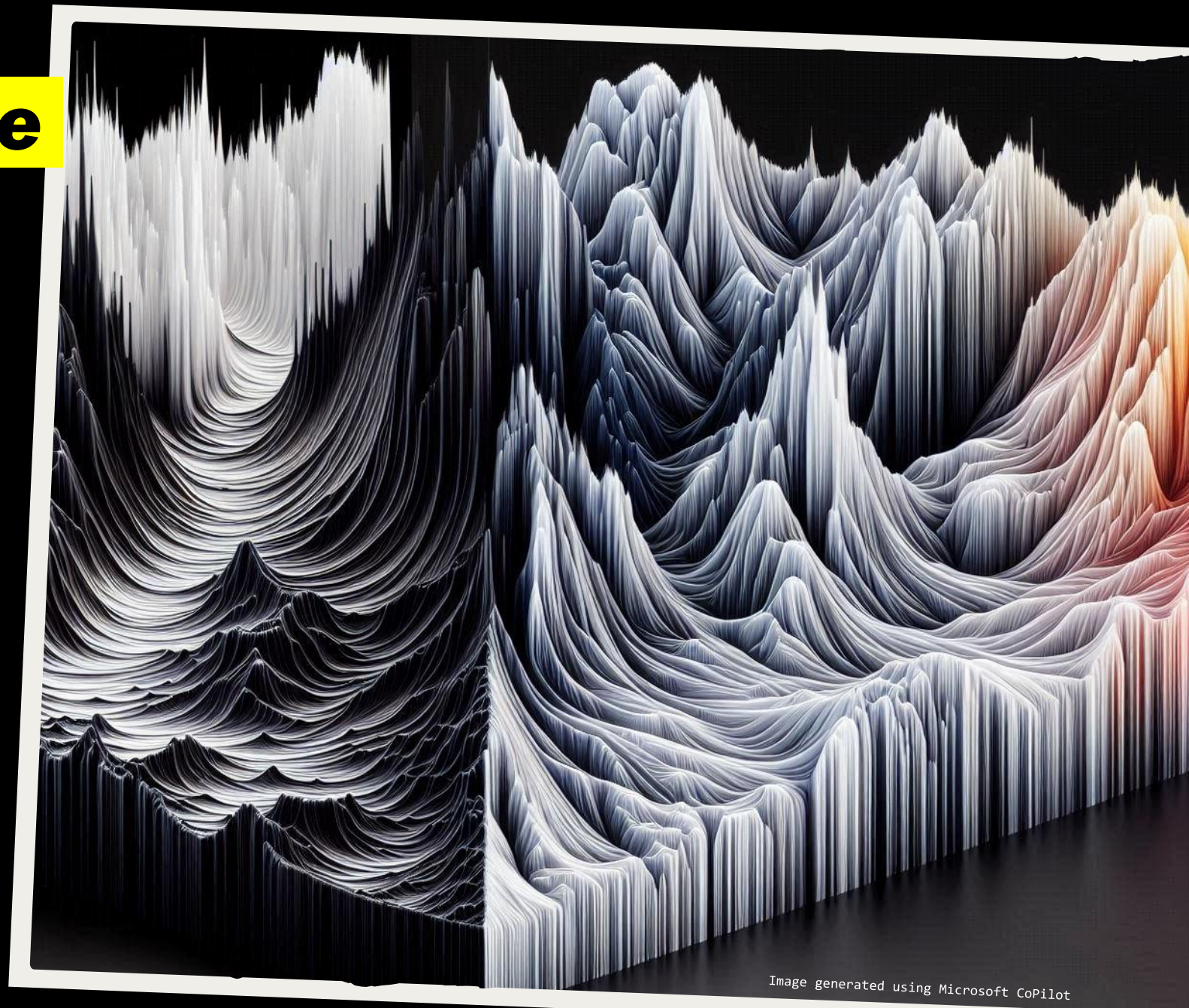
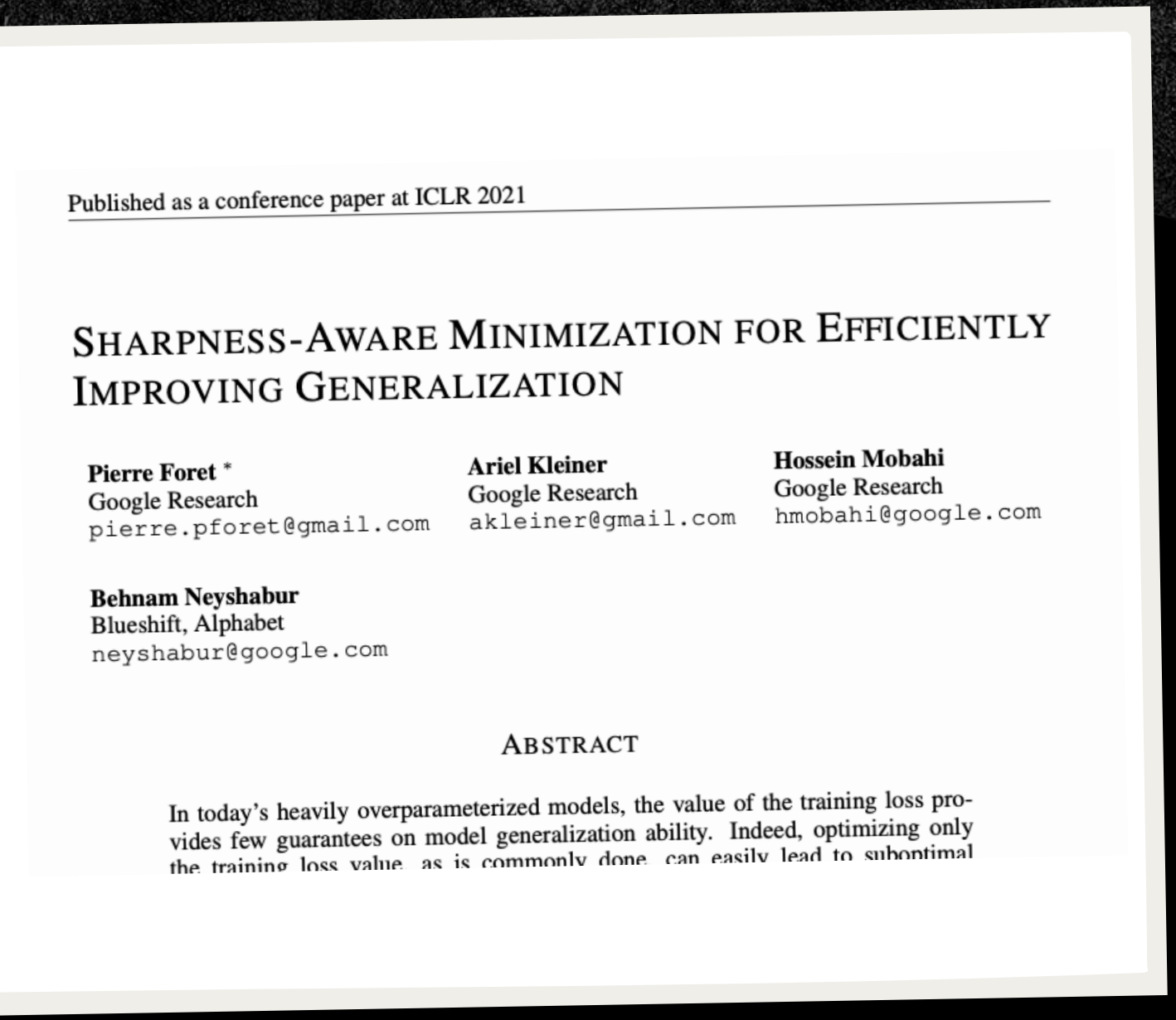
Sharpness-Aware Minimization



COURSE: MIT-5432

PROFESSOR: DR. ZHAI

AUTHORS: NICK PRANSKE,
ANDREW SOLTIS,
REID RORICK



Published as a conference paper at ICLR 2021

SHARPNESS-AWARE MINIMIZATION FOR EFFICIENTLY IMPROVING GENERALIZATION

Pierre Foret *
Google Research
pierre.pforet@gmail.com

Ariel Kleiner
Google Research
akleiner@gmail.com

Hossein Mobahi
Google Research
hmobahi@google.com

Behnam Neyshabur
Blueshift, Alphabet
neyshabur@google.com

ABSTRACT

In today's heavily overparameterized models, the value of the training loss provides few guarantees on model generalization ability. Indeed, optimizing only the training loss value, as is commonly done, can easily lead to suboptimal

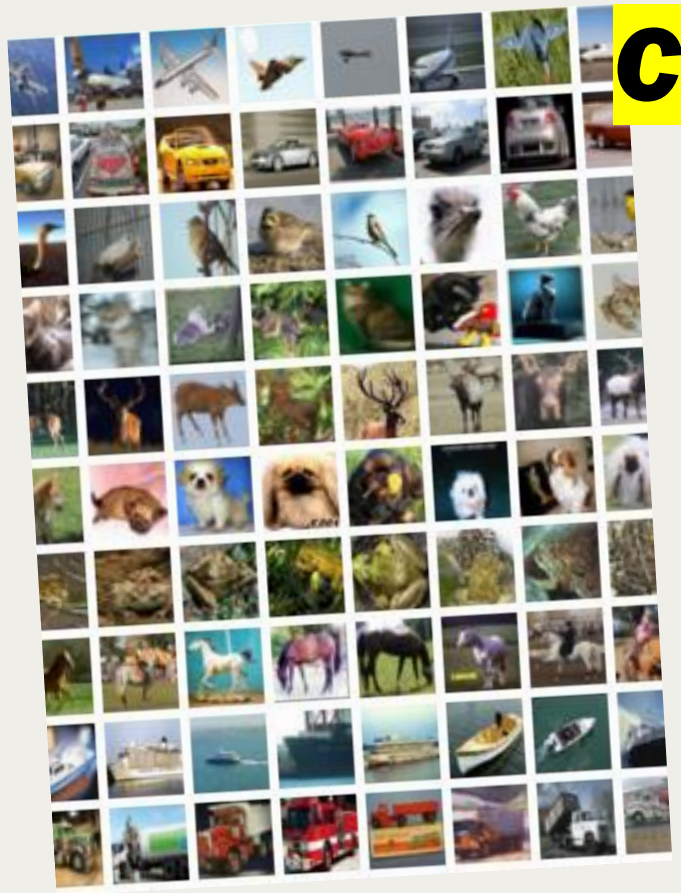
Paper

- Sharpness-Aware Minimization For Efficiently Improving Generalization

  + Google Research

- International Conference on Learning Representations 2021

CIFAR-10 Dataset



Images are from the CIFAR-10 Dataset

New dataset from the Project Proposal	Original: Street View House Numbers (SVHN) New: CIFAR-10
60,000 32x32 color images	10 classes: 6,000 images each airplane, automobile, bird, cat, deer, dog, frog, horse, ship, truck
Dataset Split	50,000 training images 10,000 validation images
Divided into Batches of 10,000	5 training batches 1 test batch

Optimization vs Model

Realized paper was related to an Optimization and not a model

Model

- Mathematical model that learns from data
- Used to make predictions or decisions
- Effectively creates the Loss Function

Optimization

- Process of fine-tuning a model
- Improves model’s performance and accuracy
- Typically aims to minimize the Loss Function

Image generated using Microsoft CoPilot

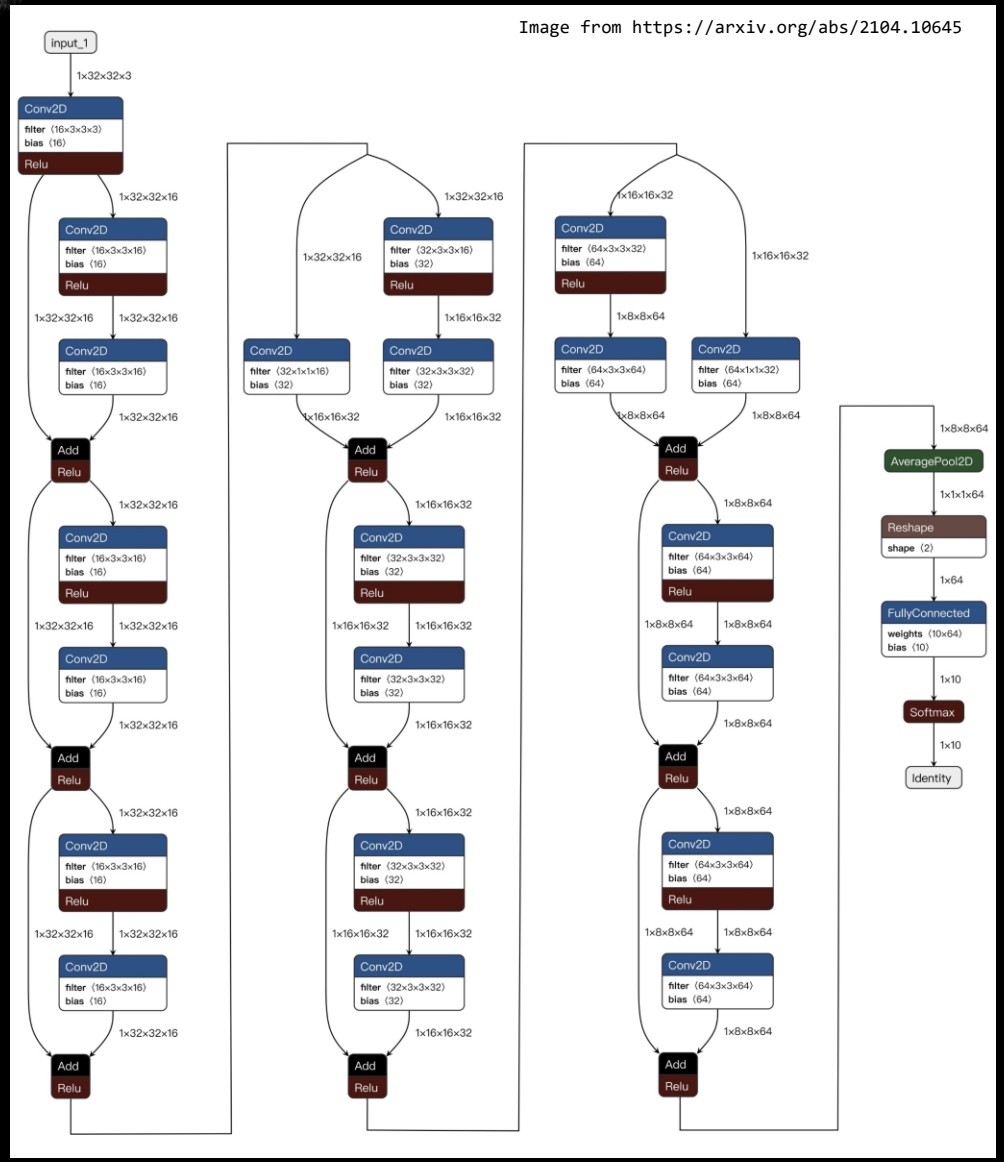
Versatility

Can be applied to a variety of models including:

- WRN-28-10†
- Shake-Shake†
- PyramidNet†
- ResNet-50†
- ResNet-101†
- ResNet-152†
- CNN36
- CNN39
- EffNet-L2 (SAM)
- EfficientNet-L2-475 (SAM)

ResNet-20

- ResNet-20, a Convolutional Neural Network (CNN) variant, forms part of the Residual Network (ResNet) family.
- Designed to tackle the vanishing gradient problem, this model enables efficient training of deep networks.
- Unique ‘skip connections’ bypass certain layers during training, enhancing learning efficiency.
- Several blocks form the architecture, each containing a set of convolutional layers, batch normalization, and ReLU activation functions.
- Learning the ‘residual’ or difference between input and output of layers is simplified through skip connections.

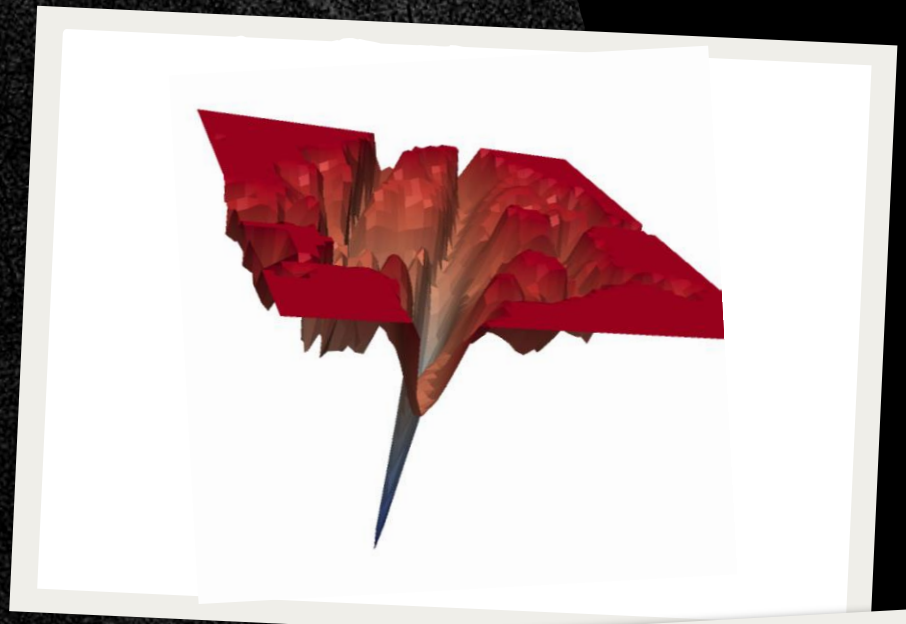


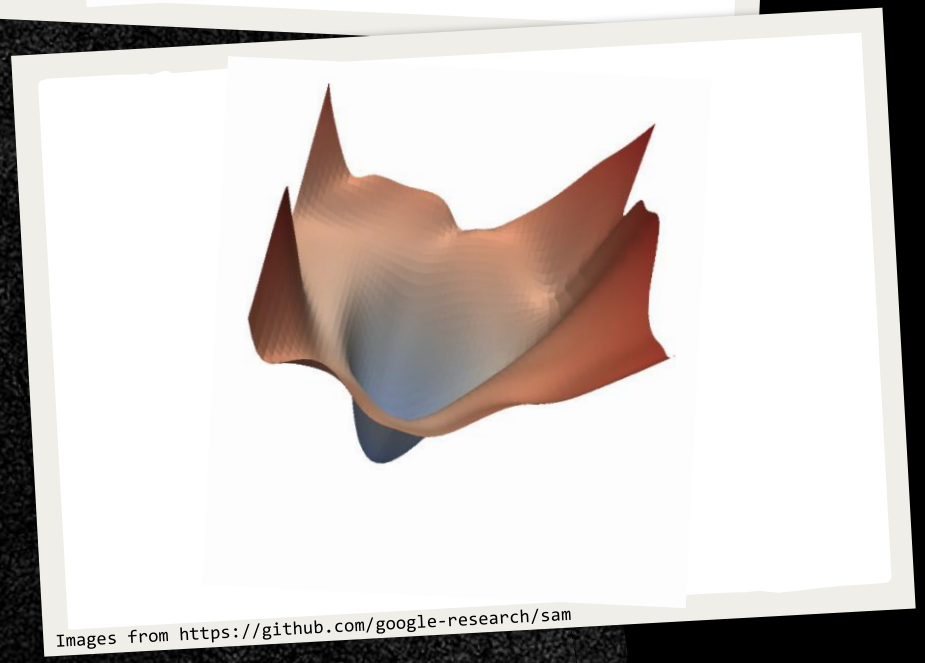
SAM

Input: Training set $\mathcal{S} \triangleq \cup_{i=1}^{n}\{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}$, Loss function $l : \mathcal{W} \times \mathcal{X} \times \mathcal{Y} \rightarrow \mathbb{R}_+$, Batch size b, Step size $\eta >$ Neighborhood size $\rho > 0$.
Output: Model trained with SAM
Initialize weights $\boldsymbol{w}_0$, $t = 0$;
while *not converged* **do**
  Sample batch $\mathcal{B} = \{(\boldsymbol{x}_1, \boldsymbol{y}_1), ...(\boldsymbol{x}_b, \boldsymbol{y}_b)\}$;
  Compute gradient $\nabla_{\boldsymbol{w}} L_{\mathcal{B}}(\boldsymbol{w})$ of the batch's training los
  Compute $\hat{\boldsymbol{\epsilon}}(\boldsymbol{w})$ per equation 2;
  Compute gradient approximation for the SAM objective (equation 3): $\boldsymbol{g} = \nabla_{\boldsymbol{w}} L_{\mathcal{B}}(\boldsymbol{w})|_{\boldsymbol{w}+\hat{\boldsymbol{\epsilon}}(\boldsymbol{w})}$;
  Update weights: $\boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \eta \boldsymbol{g}$;
  $t = t + 1$;
end
return $\boldsymbol{w}_t$

Image from https://openreview.net/pdf?id=6Tm1mposlrM

- **Objective:**
  - + SAM's goal is to minimize the empirical risk (the error on the training data) and ensure robustness (stability) against small perturbations (changes) in the parameter space (the values that define the model).
- **Method:**
  - + SAM does this by aiming to minimize both Loss and Sharpness simultaneously. This results in a higher loss value, but also a higher chance to realistically reach that loss minimum due to the decreased sharpness. This, in turn, outputs a more accurate model.





Images from https://github.com/google-research/sam

SAM

- **Solution**:
  + SAM solves the problem using a variant of gradient descent (a method to find the minimum of a function), ensuring stability during training and enhanced model generalization.
- **Compatibility**:
  + SAM can be implemented with two backpropagation steps per iteration, making it compatible with common deep-learning libraries.

SAM Analogy

Imagine you’re on a hiking adventure:

+ **Traditional Optimization**: It’s like a hiker who rushes to the top along the steepest path. Fast, but risky. The hiker may or may not actually make it to the top

+ **SAM**: It’s like a careful hiker who ensures the path is not only leading to the top but is also safe and stable. This way, the hiker is more likely to reach the summit

Code Used

- SAM Implementation by Sayak Paul – ML @ Hugging Face
  + Based on Sharpness-Aware Minimization For Efficiently Improving Generalization
- Overview of the code structure
  + Clones a GitHub repository.
  + Sets up the environment for TPUs or GPUs.
  + Loads and preprocesses the CIFAR-10 dataset.
  + Defines a custom SAMModel class.
  + Trains two models ResNet-20 and ResNet-20 w/ SAM.
- Highlights from the implementation
  + SAM Implementation: A custom train_step method is implemented within the SAMModel class to apply SAM during training. This involves a unique two-step update process.
  + Parameter Updates: In the two-step update process, parameters are first moved along the gradient direction, then updated in the opposite direction based on a second gradient calculation.

```python
# summarize history for accuracy
plt.figure(figsize=(12, 10))
plt.plot(history.history['accuracy'], 'o-', color='lime', linewidth=2, markersize=10, label='Train Accuracy')
plt.plot(history.history['val_accuracy'], 's-', color='cyan', linewidth=2, markersize=10, label='Validation Accuracy')
plt.title('Model Accuracy', fontsize=24)
plt.ylabel('Accuracy', fontsize=18)
plt.xlabel('Epoch', fontsize=18)
plt.legend(loc='lower right', fontsize=14)
plt.grid(True)
plt.show()

# summarize history for loss
plt.figure(figsize=(12, 10))
plt.plot(history.history['loss'], 'o-', color='yellow', linewidth=2, markersize=10, label='Train Loss')
plt.plot(history.history['val_loss'], 's-', color='magenta', linewidth=2, markersize=10, label='Validation Loss')
plt.title('Model Loss', fontsize=24)
plt.ylabel('Loss', fontsize=18)
plt.xlabel('Epoch', fontsize=18)
plt.legend(loc='upper right', fontsize=14)
plt.grid(True)
plt.show()

# summarize all history in one graph
fig, ax1 = plt.subplots(figsize=(12, 10))
color_acc = 'lime'
```

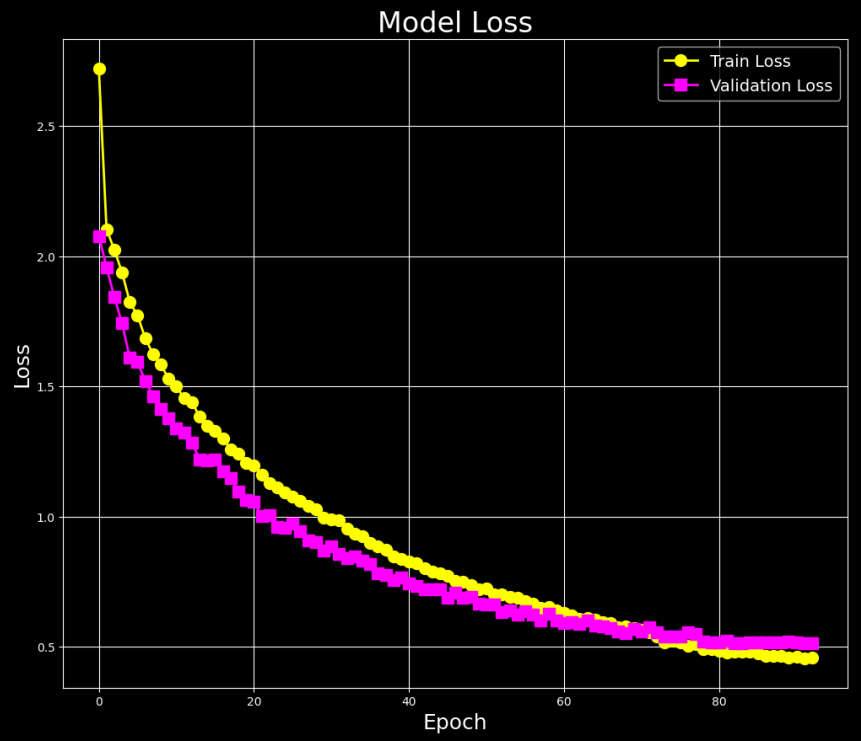
```python
# Evaluate the model on the test dataset
loss, accuracy = model.evaluate(test_ds)

# Print the accuracy
print(f"Accuracy: {accuracy*100:.2f}%")
print(f"Loss: {loss*100:.2f}%")

# Calculate the error rate
error_rate = 1 - accuracy

# Print the error rate
print(f"Error rate of the model: {error_rate*100:.2f}%")
```

Code Modifications

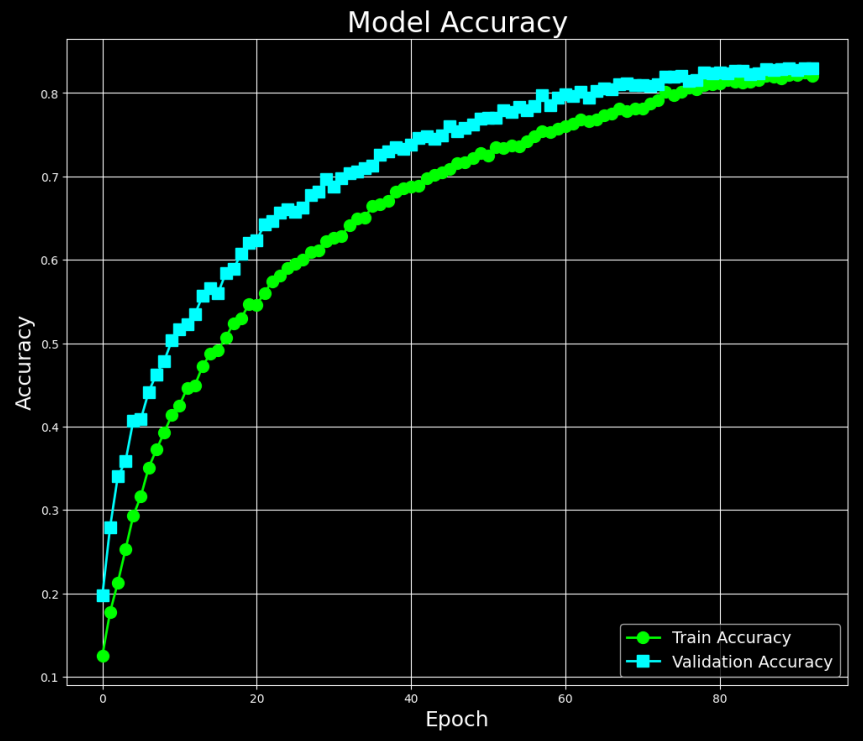
- Plots

  + Split Accuracy and Loss into two visuals

- Added Accuracy, Loss, and Error Rate



Loss: 52.27%

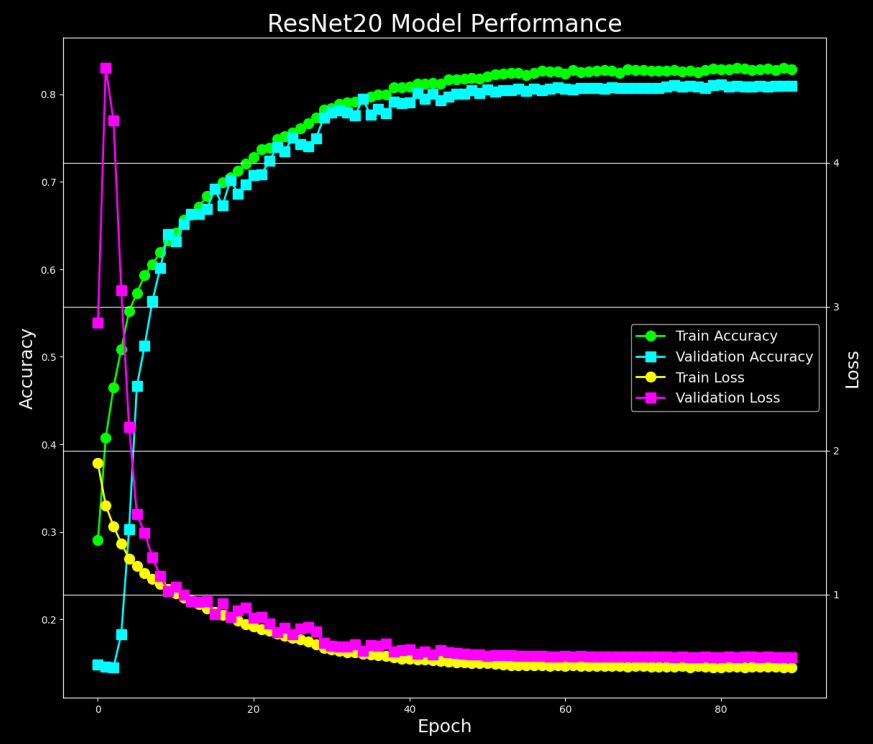


Accuracy: 82.67%

SAM Results

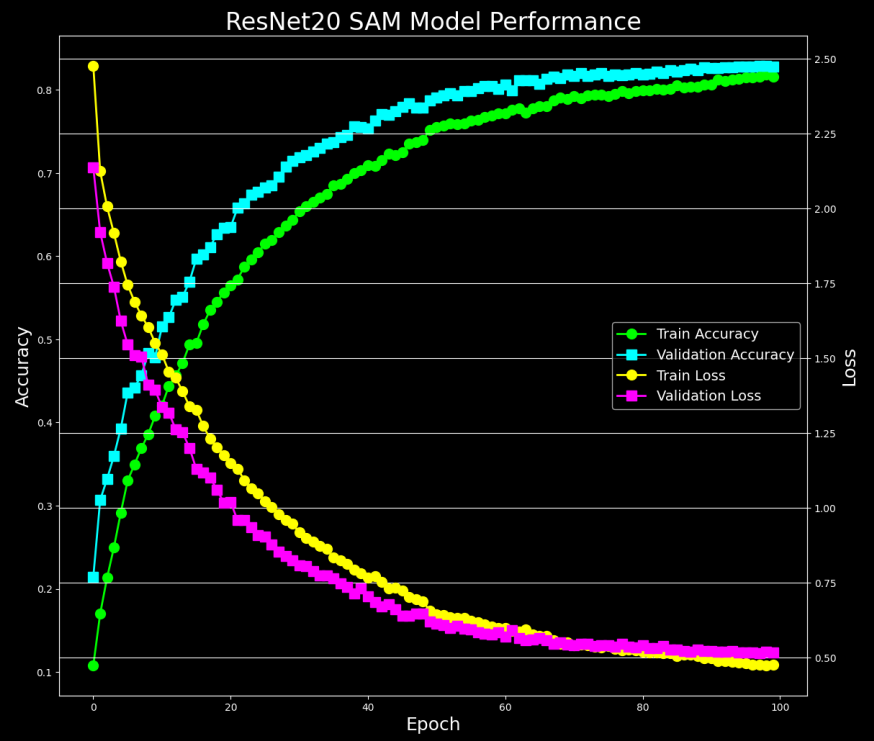
Validation performed better than training.

Without SAM



Error Rate: 18.91%

With SAM



Error Rate: 17.33%

Model Comparison

ResNet-20 model with SAM (right) performed better than the ResNet model without SAM (left).

Code Compared to Paper

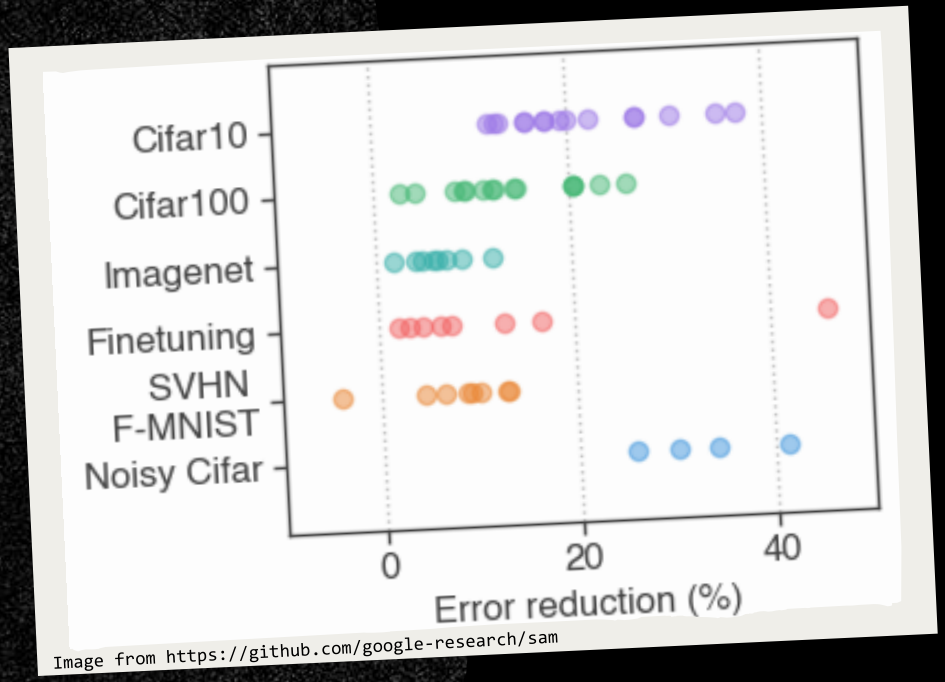


Image from https://github.com/google-research/sam

- CIFAR-10 dataset was used in addition to CIFAR-100, and ImageNet.
- ResNet-20 was not used, rather multiple other models were used.
- Both the original report and code show the same result, that models with SAM perform better than models without SAM.
- Results from the paper, each point represents a different dataset, model, data augmentation.

Conclusion

- Machine Learning Models focus on learning from data to make predictions, while Optimization is the process of fine-tuning these models for the best performance.
- Models with SAM performed better than models without SAM
  + Means SAM optimization is a good optimization
- The model performs well not just on the training data, but also on unseen data.
  + Validation performed better than Training
  + Does better on the closed-book test than the open-book test.
- SAM provides a more stable and optimal path during training.


Questions

Citations

- https://openreview.net/pdf?id=6Tm1mposlrM
- https://www.cs.toronto.edu/~kriz/cifar.html
- https://arxiv.org/abs/2104.10645
- https://colab.research.google.com/github/sayakpaul/Sharpness-Aware-Minimization-TensorFlow/blob/main/SAM.ipynb
- https://github.com/sayakpaul/Sharpness-Aware-Minim
- https://www.linkedin.com/in/yousef-elgodamy-1272b4196/