PUC Minas

Aprimorando o Desempenho da SqueezeNet na Classificação de Imagens

Tópicos em Engenharia de Software

Professor: Leonardo Vilela Cardoso

Integrantes:

- Marcos Felipe.
- Murilo Costa.
- Davi Santos.

Sumário

1. Introdução

- A SqueezeNet é uma arquitetura eficiente, com baixo número de parâmetros, ideal para dispositivos com recursos limitados.

- Contudo, seu desempenho em datasets mais complexos pode ser limitado, como em imagens com alta resolução e fundos detalhados.

- Este trabalho propõe otimizações para melhorar a acurácia da SqueezeNet nesses cenários, mantendo sua eficiência.

2. Objetivo

- Avaliar diferentes estratégias de otimização para a SqueezeNet em tarefas de classificação de imagens.

- Identificar quais modificações melhoram o desempenho do modelo de forma mais eficaz.

- Propor uma abordagem sinérgica entre arquitetura e protocolo de treinamento.

Materiais e Métodos

3.1 Ambiente de Execução

- Plataforma: Google Colab com GPU T4.
- Ferramentas: Python 3.9, PyTorch, Torchvision, NumPy, Matplotlib.

3.2 Dataset

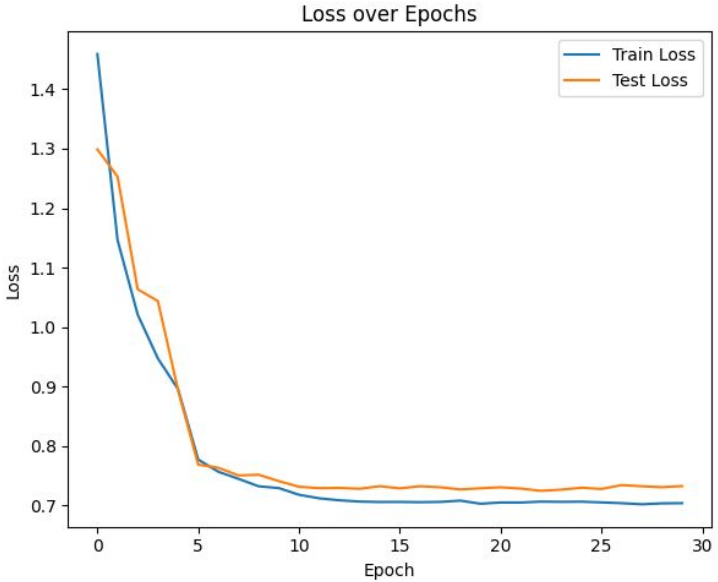
- Dataset utilizado: CIFAR-10 (60.000 imagens, 10 classes).
- Base para comparação do desempenho nas diferentes etapas.

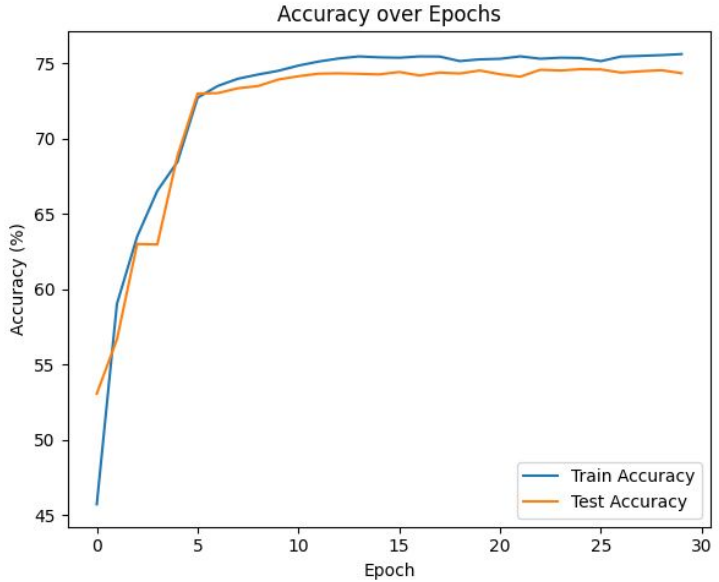
3.3 Aplicação de um processo experimental iterativo com cinco etapas:

1. Modelo base.
2. Otimizações de treinamento e dados.
3. Inserção de *SimpleAttention*.
4. Inclusão de conexões residuais.
5. Otimização holística combinando todas as melhorias.

3.3.1 Modelo Base

- Squeezenet com Early Stop(Parar no caso de Overfitting)





3.3.1 Modelo Base

- Squeezenet com Early Stop(Parar no caso de Overfitting)

3.3.2 Otimizações de Treinamento e Dados

- Aumento de dados (data augmentation) para maior variabilidade.
- Fine-tuning para ajustar o modelo a novos dados.
- Conversão de imagens em RGB para padronização.

3.3.2 Otimizações de Treinamento e Dados

```
Acurácia por classe:
plane: 83.42%
car: 91.26%
bird: 72.56%
cat: 81.20%
deer: 77.28%
dog: 82.24%
frog: 83.22%
horse: 82.18%
ship: 90.04%
truck: 88.90%
Test Loss: 0.572 | Test Acc: 80.84%
EarlyStopping counter: 1 out of 10
```

```
Acurácia por classe:
plane: 84.52%
car: 92.20%
bird: 74.32%
cat: 83.10%
deer: 78.86%
dog: 84.28%
frog: 84.50%
horse: 83.22%
ship: 91.90%
truck: 90.06%
Test Loss: 0.542 | Test Acc: 81.77%
EarlyStopping counter: 10 out of 10
Early stopping triggered
Melhor acurácia no teste: 81.90%
```

Predição: ship
Prob: 100.0%

Predição: ship
Prob: 97.4%

Predição: car
Prob: 96.0%

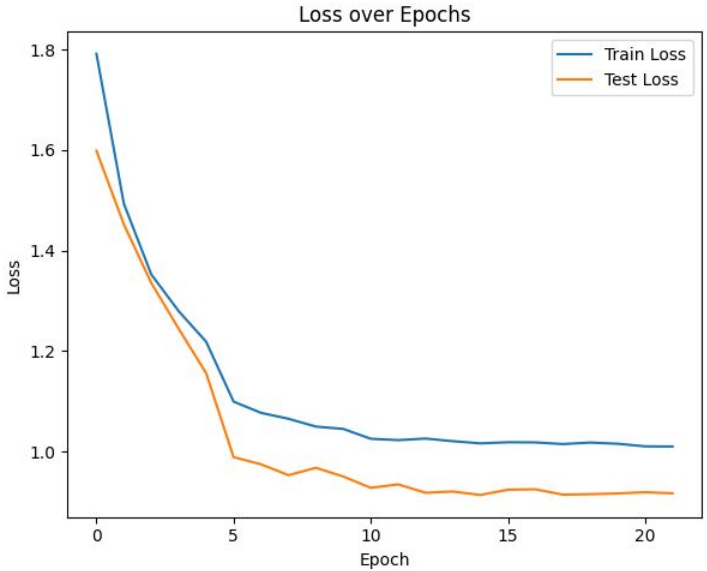
Predição: cat
Prob: 91.8%

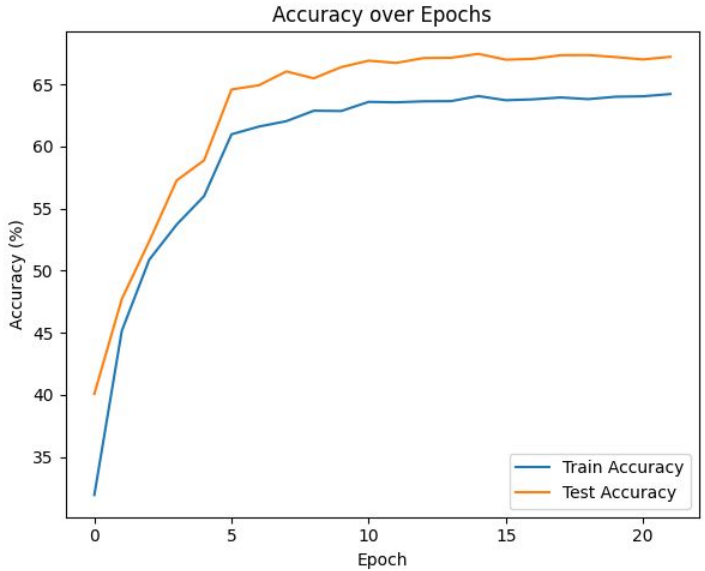
Predição: ship
Prob: 58.6%

Predição: cat
Prob: 85.6%

3.3.3 Inserção de *SimpleAttention*

- Módulos de atenção: SimpleAttention e ChannelAttention.



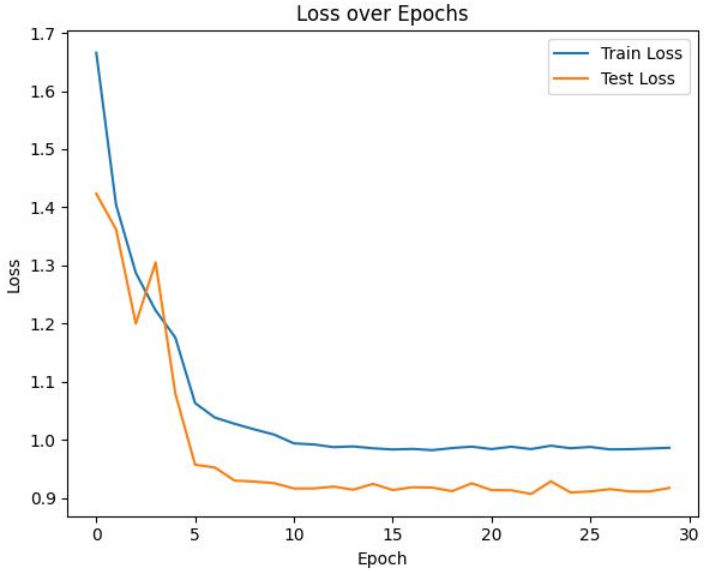


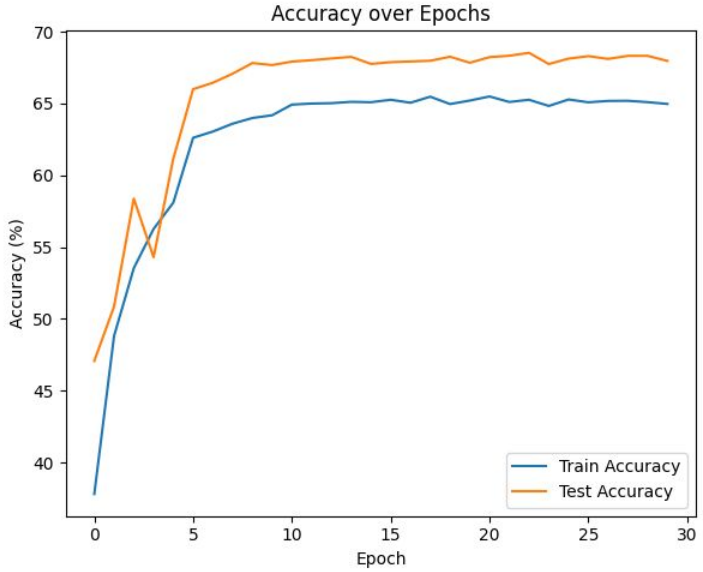
3.3.3 Inserção de *SimpleAttention*

- Módulos de atenção: SimpleAttention e ChannelAttention.

3.3.4 Inclusão de conexões residuais

- Módulos residuais da arquitetura ResNet.
- Dropout no fim das convoluções.





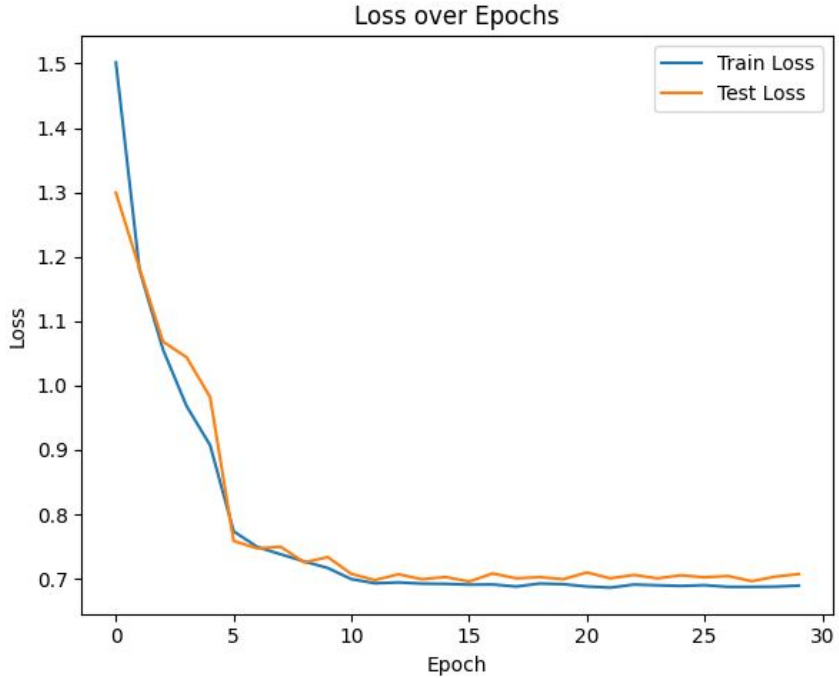
3.3.4 Inclusão de conexões residuais

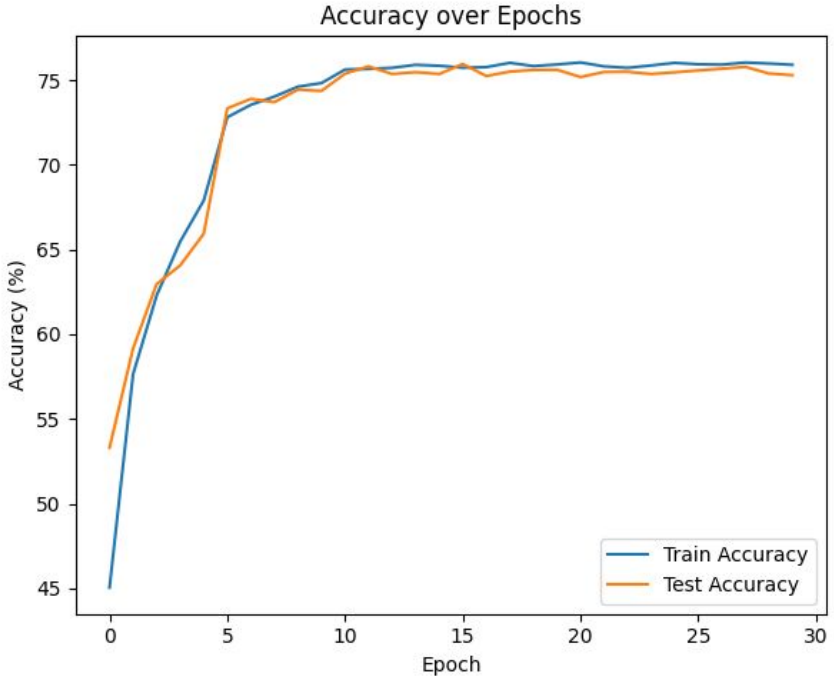
- Módulos residuais da arquitetura ResNet.
- Dropout no fim das convoluções.

3.3.5 Otimização holística combinando todas as melhorias

- Arquitetura otimizada com blocos melhorados.
- Data augmentation avançado.
- OneCycleLR scheduler.
- Label smoothing.
- Early stopping.
- Inicialização de pesos adequada

3.3.5 Otimização holística combinando todas as melhorias





3.3.5 Otimização holística combinando todas as melhorias

4. Resultados

Etapa Experimental	**Principais Modificações**	**Acurácia de Teste (%)**
1. Modelo Base	SqueezeNet Original	74,09
2. Otimização Inicial	Fine-Tuning + Aumento de Dados	81,90
3. Atenção Simples	Adição de Módulo *SimpleAttention*	67,23
4. Conexões Residuais	Adição de Blocos Residuais à arquitetura com atenção	67,96
5. Otimização Final	Arquitetura e Treinamento Avançados (Abordagem Holística)	**86,86**

5. Conclusões

- Aumento de dados e fine-tuning são altamente eficazes para ganhos iniciais.
- Modificações arquiteturais isoladas podem reduzir a acurácia.
- A abordagem holística, que combina arquitetura e treinamento, foi a única capaz de superar significativamente o modelo base.
- Conclui-se que o melhor desempenho requer uma co-otimização integrada entre arquitetura da rede e protocolo de treinamento.

8. Referências

- Y. Lecun, L. Bottou, Y. Bengio and P. Haffner, ”Gradient-based learning applied to document recognition,” in Proceedings of the IEEE, vol. 86, no. 11, pp. 2278-2324, Nov. 1998, doi: 10.1109/5.726791. keywords: Neural networks;Pattern recognition;Machine learning;Optical character recognition software;Character recognition;Feature extraction;Multi-layer neural network;Optical computing;Hidden Markov models;Principal component analysis.
- Iandola, Forrest N., et al. ”SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and¡ 0.5 MB model size.” arXivpreprintarXiv:1602.07360 (2016).
- A. Krizhevsky, ”Learning Multiple Layers of Features from Tiny Images,” Tech. Rep., University of Toronto, Toronto, ON, Canada, 2009.
- Targ, Sasha, Diogo Almeida, and Kevin Lyman. ”Resnet in resnet: Generalizing residual architectures.” arXiv preprint arXiv:1603.08029 (2016).

Perguntas?