Cattle Breed Classification Using CNNs

A Comparative Study of ResNet18 and VGG16 with Transfer Learning

Samira Ismayilova, Agayeva Firuza, Gallacov Emil
Group CE 22.2, Baku Higher Oil School

December 2025

Abstract. In this project we implement and compare two convolutional neural network (CNN) architectures, ResNet18 and VGG16, for multi–class cattle breed classification. Both networks are trained using transfer learning with two different optimizers (SGD and Adam). In addition, a scratch ResNet18 model is trained for baseline comparison. We use TensorBoard to monitor training and validation accuracy and loss, evaluate the models on a held–out test set, and compute macro F1–scores and class–wise accuracies. The experiments show that transfer learning significantly improves performance and that ResNet18 with SGD achieves the best overall results on the 50–class dataset.

1 Introduction

Automatic cattle breed recognition is an important problem in smart farming, livestock monitoring and breeding management. Manually identifying breeds from images is slow and subjective, especially when many visually similar breeds must be distinguished. Deep learning based image classification provides a scalable and objective solution.

Convolutional Neural Networks (CNNs) are the current state–of–the–art models for image recognition. However, training deep CNNs from scratch requires large labelled datasets and considerable computational resources. Transfer learning mitigates this problem by reusing feature extractors that were pretrained on large datasets such as ImageNet.

The aim of this project is to:

- Implement two CNN architectures: ResNet18 and VGG16.
- Train and compare them with two optimizers: SGD and Adam.
- Study the effect of transfer learning versus training from scratch.
- Analyse training behaviour via TensorBoard accuracy and loss curves.

2 Dataset and Preprocessing

The dataset contains images of approximately 50 different cattle breeds. The data are divided into three disjoint subsets:

- **Training set:** 5949 images
- **Validation set:** 1254 images
- **Test set:** 1328 images

All images are resized to 224×224 pixels to match the standard input size of ResNet18 and VGG16. For the training set we apply basic data augmentation:

- random horizontal flip,
- random rotation (up to $\pm 15°$).

Afterwards all images are converted to tensors and normalized using ImageNet mean and standard deviation. Validation and test sets are only resized and normalized, without augmentation.

3 Model Architectures

3.1 ResNet18

ResNet18 is a residual network with identity skip connections that help gradients flow through the network and allow deeper models to be trained. In this project we use the ImageNet–pretrained ResNet18 implementation from PyTorch. The final fully–connected layer is replaced by a new linear layer with 50 outputs, corresponding to the number of cattle breeds. We consider two versions:

- **ResNet18 (TL):** transfer learning, starting from ImageNet weights.
- **ResNet18 (Scratch):** the same architecture initialized randomly.

3.2 VGG16

VGG16 is a deeper CNN with 16 weight layers organised in blocks of 3×3 convolutions followed by max pooling. It has more parameters than ResNet18 and acts as a strong generic feature extractor. We use an ImageNet–pretrained VGG16 model and freeze the convolutional feature extractor. Only the classifier part (fully–connected layers) is fine–tuned on the cattle dataset.

4 Training Setup

All experiments are implemented in PyTorch and executed on Google Colab with GPU acceleration. The common settings are:

- loss function: cross–entropy,
- batch size: 32,
- ResNet18 (transfer learning): 5 epochs,
- ResNet18 (scratch): 3 epochs,
- VGG16 (transfer learning): 1 epoch (due to model size and runtime),
- optimizers:
  - SGD with learning rate 0.001 and momentum 0.9,
  - Adam with learning rate 0.0005.
- monitoring: TensorBoard for accuracy and loss curves.

For transfer learning experiments the convolutional feature extractor is kept frozen, and only the final classifier layers are updated. The best model weights with respect to validation accuracy are saved and later evaluated on the test set. Trained models are stored in `.pth` format in Google Drive for future reuse.

5 Quantitative Results

Table 1 summarises the test performance of all models. In addition to overall accuracy we report the macro F1–score, which averages per–class F1 and is more informative under class imbalance.

Table 1: Comparison of CNN models on the cattle breed classification task.

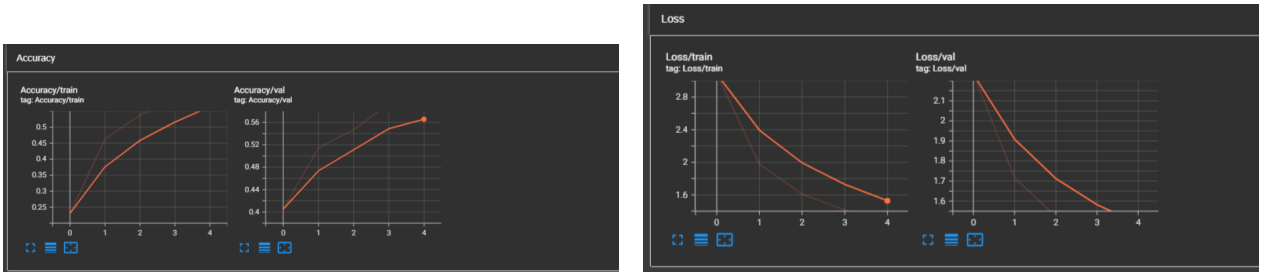
Model	Pretrained	Optimizer	Test Accuracy	Test F1
ResNet18	Yes	SGD	0.5813	0.5319
ResNet18	Yes	Adam	0.4902	0.4455
ResNet18	No	SGD	0.1762	0.0991
VGG16	Yes	SGD	0.4495	0.3820
VGG16 VGG16	Yes	Adam	0.3742	0.3560

ResNet18 with transfer learning and SGD clearly achieves the best overall performance. Training ResNet18 from scratch performs very poorly, which confirms that transfer learning is essential for this dataset.

6 TensorBoard Accuracy and Loss Curves

In this section we present the TensorBoard plots for each model–optimizer combination. Every figure shows both training and validation curves and helps to interpret convergence, overfitting behaviour and optimisation stability.

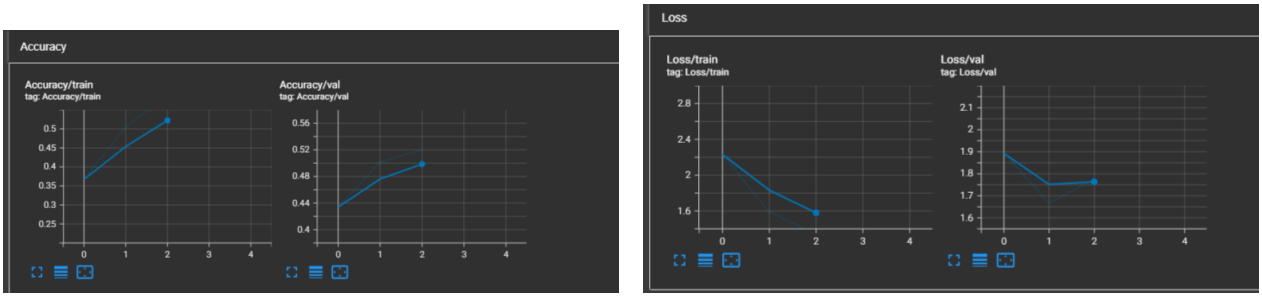
6.1 ResNet18 with SGD



(a) Accuracy (train vs validation). (b) Loss (train vs validation).

Figure 1: ResNet18 (transfer learning, SGD). The validation accuracy steadily increases up to around 59%, while the loss decreases smoothly, indicating stable training and good generalisation.

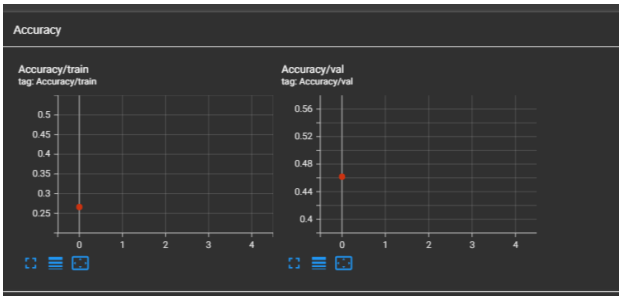
6.2 ResNet18 with Adam



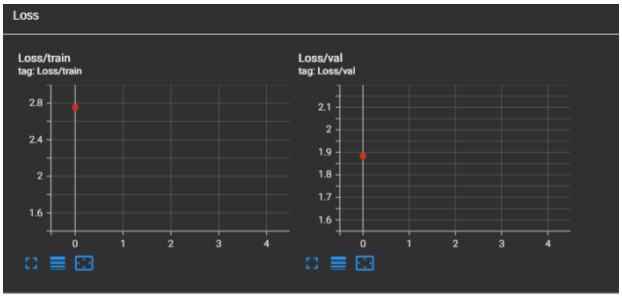
(a) Accuracy (train vs validation). (b) Loss (train vs validation).

Figure 2: ResNet18 (transfer learning, Adam). Adam converges faster in the first epochs, but the validation accuracy saturates below the SGD version and the gap between training and validation curves is slightly larger, which explains the lower test performance.

6.3 VGG16 with SGD



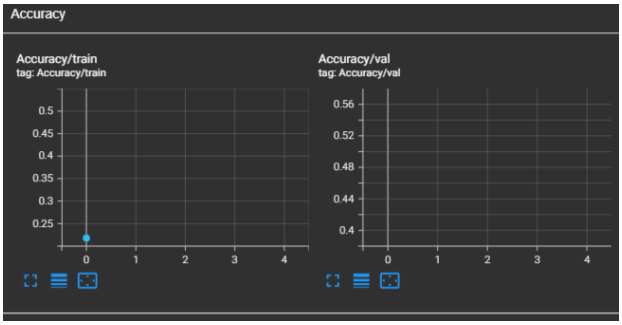
(a) Accuracy (train vs validation).



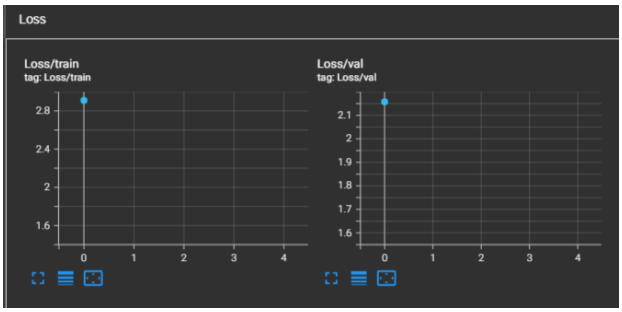
(b) Loss (train vs validation).

Figure 3: VGG16 (transfer learning, SGD). With only one epoch both accuracy and loss curves indicate that the model is still improving and has not fully converged yet. More epochs would likely lead to higher accuracy.

6.4 VGG16 with Adam



(a) Accuracy (train vs validation).



(b) Loss (train vs validation).

Figure 4: VGG16 (transfer learning, Adam). The curves show more fluctuation, reflecting the combination of a very deep model, limited epochs and the adaptive optimiser. Nevertheless, the model still achieves reasonable accuracy for a single training epoch.

7 Discussion

The experiments confirm several expected behaviours:

- **Transfer learning vs. scratch.** The scratch ResNet18 model achieves only 17.6% accuracy and almost zero macro F1, while the transfer learning variants perform much better. This shows that starting from ImageNet features is crucial when the dataset is relatively small.
- **ResNet18 vs. VGG16.** ResNet18 consistently outperforms VGG16 in both accuracy and F1–score. Its residual connections enable easier optimisation, while VGG16 is heavier and needs more epochs and regularisation.
- **SGD vs. Adam.** In our setting SGD with momentum generalises better than Adam. Adam converges quickly but may overfit more; this is visible in the validation curves and final test metrics.
- **Class imbalance.** Class–wise accuracies (not fully listed here) show that some breeds are recognised with very high accuracy, while others are much harder. Techniques such as weighted loss, resampling or focal loss could further improve the macro F1–score.

8 Conclusion

In this project we implemented a complete deep learning pipeline for cattle breed classification using PyTorch, Google Colab and TensorBoard. Two CNN architectures, ResNet18 and VGG16, were trained with different optimisers and compared under the same experimental conditions.

The best performing configuration is ResNet18 with transfer learning and SGD, reaching 58.13% test accuracy and a macro F1–score of 0.5319. The results clearly highlight the benefits of transfer learning and show that architecture and optimiser choices strongly affect final performance.

Future work may include:

- running longer training schedules for VGG16,
- applying class–balanced or focal loss functions,
- trying more advanced architectures such as EfficientNet or Vision Transformers.

References

[1] K. He, X. Zhang, S. Ren, and J. Sun, *Deep Residual Learning for Image Recognition*, CVPR, 2016.

[2] K. Simonyan and A. Zisserman, *Very Deep Convolutional Networks for Large-Scale Image Recognition*, arXiv:1409.1556, 2014.

[3] PyTorch Documentation, https://pytorch.org