Assignment 3 - Technical Report
Yong Zhu Cheng A0275768H

A) Model Overview

The following is a high-level comparison of the models tested for this assignment. As per the assignment instructions, no pre-trained weights were used at any point in the project:

Model	Accuracy	Precision	Recall	F1
CNN	0.617	0.638	0.617	0.607
Resnet18	0.743	0.741	0.743	0.737
Resnet34	0.613	0.648	0.613	0.600
Resnet34 (Custom)	0.569	0.590	0.579	0.579
Resnet50	0.679	0.699	0.679	0.675
ResNext	0.608	0.617	0.608	0.608
AlexNet	0.607	0.600	0.607	0.597

* Metrics are computed using validation set. Test set result for final model is given in the Conclusion.
**Key hyperparameters, e.g. learning rate and batch size, are unique to each model due to their differing characteristics

The CNN and Resnet (KIV) architectures were built from scratch, with model attributes and hyperparameters tuned for optimal performance. The other architectures were drawn directly from the torchvision library, with minor adjustments to accommodate single-channel inputs.

From the above results, we clearly observe that Resnet18 yielded the best performance, with superior results across all metrics.

B) Data Processing

The images in the dataset exhibit the following characteristics:

- Balanced across classes
- Irregularly sized
- Generally small and low-resolution

The absence of class imbalance suggests that there is no pertinent need to account for that through different metrics (e.g. macro-averaged metrics). The latter two observations, in turn, led me to experiment with the following methods of data processing and augmentation by leveraging the torchvision package:

1) For the base case, resize images to the lowest common denominator - (200,200)
2) Randomly flip images in horizontal direction
3) Upsize images via up-sampling and bicubic interpolation (Reference)
4) Randomly rotate images by some small degree.
5) Randomly resize and crop images.
6) Randomly perform affine transformation on images

For #2, the hypothesis is that with the smaller size and resolution of the images, it might help to enlarge them so that smaller features can be captured more clearly. As for #4-6, the intuition is that these transformations serve to mimic different perspectives and angles by which a scene is captured, thereby improving the robustness of the features identified. Using the ResNet18 model as control, I tested the transformations and observed the following results:

- The transformations yielding the best improvements are horizontal flipping and rotation.
    - However, large rotations have a opposite effect, while angles smaller than 30 seem to fare poorly.
- Cropping, affine transformation and up-sampling reduced model performance.

It is possible that cropping and affine transformation might have generated an amount of noise that was not helpful for model learning, especially when the distortions are beyond what one would observe in a typical photo (like the ones in the validation and test sets). For example, a picture that zooms specifically into the bedsheets in a bedroom might look quite amorphous, making it hard for the machine to clearly recognize that it in fact depicts a bedroom. Additionally, it is plausible that up-sampling did not improve predictions as the effective resolution of each image is no higher than before. While there are existing super-resolution models, implementing them as a preparation step could be a scope for future learning and research. Furthermore, I have experimented with duplicating the images instead of transforming them in place, i.e. training the model on 2-3 variants of the same image, but this did not improve the model, which is intriguing.

C) Training Methodology

In addition to data augmentation, the following steps were taken in the training process:

- The dataset was split on a 64-16-20 (Train-Val-Test) basis for training and evaluation.
- Early stopping with a patience of 20 epochs, for regularization purposes.
- Use of the Adam optimizer as well as a learning rate scheduler to adjust the learning rate according to training progress. Accordingly, the step size and gamma value of the scheduler were tuned as hyperparameters.
- Random sampling was incorporated into the data loader to increase the reliability of the validation results.
- Model checkpoints were saved for ease of troubleshooting, to check for issues like vanishing, exploding or null gradients.

D) Model Analysis

*Note that the listed hyperparameter combinations are not the only ones tested; many others have been applied and omitted for brevity.

<u>Basic CNN</u>

For this architecture, I built a basic CNN model consisting of n convolutional layers, each with Max Pooling, Batch Normalization and ReLU activation. I experimented with several variables, including the number of convolutional layers, the number of filters per layer, dropout inclusion and the number of attention layers added to the fully connected

layer. The intuition behind the attention layer is to capture any complex relationships between the detected features that a linear layer might not capture as well, especially if the features relate to each other in a more associative than additive way, e.g. relationships between the furniture and the wall that might suggest an office instead of a bedroom. The following are the results:

Model	Accuracy	Precision	Recall	F1
Baseline*	0.500	0.562	0.500	0.477
1 attention layer	0.508	0.510	0.508	0.499
3 layers	0.492	0.550	0.492	0.488
5 layers	0.617	0.638	0.617	0.607
5 layers w/ attention	0.533	0.575	0.533	0.535

*4 layers, without attention

We observe that the model performs better with attention than without, which may support my earlier hypothesis. Additionally, the model is very prone to overfitting, performing much poorly with a small increase in model complexity and with a marginally higher learning rate or batch size, i.e. the learning rate and batch size must be kept at a low value (not documented here for brevity).

AlexNet

The AlexNet model performs marginally better than CNN, albeit requiring a smaller learning rate, which is understandable due to the model's depth:

Learning rate	Accuracy	Precision	Recall	F1
5e-5	0.607	0.600	0.607	0.597
1e-4	0.533	0.534	0.533	0.521

With its regularization and data augmentation techniques, as well as its fine-tuned handling of the vanishing and exploding gradient problems, it is of little surprise that the model works better than a basic CNN architecture.

ResNet

The ResNet variants yielded the best performance overall by some margin. For this model type, I tried to construct a custom ResNet34 model, drawing largely from this tutorial by Nouman (2022). This allowed me to experiment with different combinations of filter/channel numbers based on the intuitions garnered from the CNN tests. Specifically, my guess was that a smaller number of filters per convolutional layer might work better, which turned out to be invalid:

Custom RNN*	Accuracy	Precision	Recall	F1
[10,20,25,25]	0.479	0.513	0.479	0.470
[20,30,40,40]	0.569	0.590	0.579	0.579

*Combination of filters across the 4 convolutional blocks. Original combination: 64,128,256,512

That said, there is perhaps a sweet spot in terms of model depth and breadth, and the findings in section A clearly suggest an overfitting issue with the deeper ResNet models. This may be attributable to the relative smallness and simplicity of the images in the dataset. Some observations are as follows:

- ResNet18 did better with a smaller learning rate than ResNet34. A hypothesis here is that the vanishing gradient problem, while partially mitigated, is not entirely solved by ResNet, and a larger learning rate might be required to offset that for the deeper ResNet model.
    - With a smaller learning rate, the gamma value and step size of the scheduler must be adjusted to prevent the learning from slowing down too quickly during training.
- Both models perform significantly better with a larger batch size, i.e. 32. This implies that the noise introduced by a smaller batch size disrupts learning, suggesting that overfitting due to model depth may be less of an issue when compared to other models. This makes sense, as in the ResNet case, regularization should easily skip over these layers if they do not contribute significantly to model performance.
- Surprisingly, ResNext did not perform quite as well as expected.

E) Conclusion

Overall, the superior performance of ResNet is not surprising and has been something of an established, even empirical fact for a while (Anwar, 2019).

Final Model	**Accuracy***
ResNet18	0.710

*On test set

For the project, another model that was implemented briefly was VGG, specifically its -11 and -16 variants. However, they were omitted from this study due to their long training time and the poor performance yielded in the few runs completed. More can be done to tune these models accordingly for a much better performance. Other possible points of improvement include experimenting with a custom class for ResNet18 to allow greater customization, as well as trying out other models like GoogLeNet.

F) References

Anwar, A. (2019, June 7). Towards Data Science. https://towardsdatascience.com/the-w3h-of-alexnet-vggnet-resnet-and-inception-7baaaecccc96

Nouman. (2022). Writing ResNet from Scratch in PyTorch. Paperspace. https://blog.paperspace.com/writing-resnet-from-scratch-in-pytorch/