DAFTAR PUSTAKA

Albawi, S., Mohammed, T. A. M., & Alzawi, S. (2017). Understanding of a convolutional neural network. In 2017 International Conference on Engineering and Technology (ICET) (pp. 1-6). IEEE.

Balai Besar Penelitian Tanaman Padi. (2015). *Pengendalian Penyakit Kresek dan Hawar Daun Bakteri*. http://bbpadi.litbang.pertanian.go.id

Bali, R., & Ghosh, T. (2018). *Hands-On Transfer Learning with Python* (S. Shetty, T. Gupta, U. Guha, S. Nikalje, & S. Editing (eds.); 1st ed.). Packt Publishing Ltd.

Barman, R., Deshpande, S., Agarwal, S., & Inamdar, U. (2019). Transfer Learning for Small Dataset. *National Conference on Machine Learning*, *March*. https://www.researchgate.net/publication/332407279_Transfer_Learning_for_Small_Dataset

BPS. (2020). Statistik Luas Panen dan Produksi Padi. *Berita Resmi Statistik*, *2*(16), 1–12. https://www.bps.go.id/pressrelease/2020/02/04/1752/

Chollet, F. (2017). Xception: Deep learning with depthwise separable convolutions. *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, *2017-Janua*, 1800–1807. https://doi.org/10.1109/CVPR.2017.195

Eka Putra, W. S. (2016). Klasifikasi Citra Menggunakan Convolutional Neural Network (CNN) pada Caltech 101. *Jurnal Teknik ITS*, *5*(1). https://doi.org/10.12962/j23373539.v5i1.15696

Ghosh, A., Sufian, A., Sultana, F., Chakrabarti, A., & De, D. (2019). Fundamental concepts of convolutional neural network. In *Intelligent Systems Reference Library* (Vol. 172, Issue June). https://doi.org/10.1007/978-3-030-32644-9_36

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, *2016-Decem*, 770–778. https://doi.org/10.1109/CVPR.2016.90

Herlisa, E. (2019). *Penyakit Bercak Daun Pada Tanaman Padi*. http://cybex.pertanian.go.id/artikel/90421/penyakit-bercak-daun-pada-tanaman-padi/

Jiang, J., Feng, X., Liu, F., Xu, Y., & Huang, H. (2019). Multi-Spectral RGB-NIR Image Classification Using Double-Channel CNN. *IEEE Access*, *7*, 20607–20613. https://doi.org/10.1109/ACCESS.2019.2896128

Kusumanto, R. D., Tompunu, A. N., & Pambudi, S. (2011). Klasifikasi Warna Menggunakan Pengolahan Model Warna HSV Abstrak. *Jurnal Ilmiah Teknik Elektro*, *2*(2), 83–87.

LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, *86*(11), 2278–2323. https://doi.org/10.1109/5.726791

Misra, S., Jeon, S., Lee, S., Managuli, R., & Kim, C. (2020). Multi-channel transfer learning of chest x-ray images for screening of covid-19. *ArXiv*, 1–7.

Narkhede, S. (2018). *Understanding AUC - ROC Curve*. https://towardsdatascience.com/understanding-auc-roc-curve-68b2303cc9c5

O'Shea, K., & Nash, R. (2015). An Introduction to Convolutional Neural Networks. *An Introduction to Convolutional Neural Networks*, 1–11. https://doi.org/10.1007/978-1-4842-5648-0

Pamungkas, A. (2017). *Pengolahan Citra*. https://pemrogramanmatlab.com/pengolahan-citra-digital/

Pan, S. J., & Yang, Q. (2010). A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, *22*(10), 1345–1359. https://doi.org/10.1109/TKDE.2009.191

Premi, M. S. G., Narmadha, R., & Bernatin, T. (2019). A Brief Survey on Diseases of Paddy Plant. *Journal Pharmaceutical Sciences and Research*, *11*(7), 2739–2743.

Rajayogi, J. R., Manjunath, G., & Shobha, G. (2019). Indian Food Image Classification with Transfer Learning. *CSITSS 2019 - 2019 4th International Conference on Computational Systems and Information Technology for Sustainable Solution, Proceedings*, *4*, 1–4. https://doi.org/10.1109/CSITSS47250.2019.9031051

Ramesh, S., & Vydeki, D. (2020). Recognition and classification of paddy leaf diseases using Optimized Deep Neural network with Jaya algorithm. *Information Processing in Agriculture*, *7*(2), 249–260. https://doi.org/10.1016/j.inpa.2019.09.002

RD. Kusumanto, A. N. T. (2011). Pengolahan Citra Digital Untuk Mendeteksi Obyek Menggunakan Pengolahan Warna Model Normalisasi. *Seminar Nasional Teknologi Informasi & Komunikasi Terapan 2011*.

Santosa, B., & Umam, A. (2018). *Data Mining dan Big Data Analytics* (Isa (ed.); 2nd ed.). Penebar Media Pustaka.

Santoso, A., & Ariyanto, G. (2018). Implementasi Deep Learning Berbasis Keras Untuk Pengenalan Wajah. *Emitor: Jurnal Teknik Elektro*, *18*(01), 15–21. https://doi.org/10.23917/emitor.v18i01.6235

Simonyan, K., & Zisserman, A. (2015). Very deep convolutional networks for large-scale image recognition. *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, 1–14.

Suyanto. (2018). *Machine Learning Tingkat Dasar dan Lanjut* (1st ed.). Informatika Bandung.

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., & Wojna, Z. (2016). Rethinking the Inception Architecture for Computer Vision. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, *2016-Decem*, 2818–2826. https://doi.org/10.1109/CVPR.2016.308

Texas Plant Disease Handbook. (n.d.). *Rice*. Retrieved February 23, 2021, from https://plantdiseasehandbook.tamu.edu/food-crops/cereal-crops/rice/

Tyagi, V. (2018). Understanding Digital Image Processing. *Understanding Digital Image Processing*, *November*. https://doi.org/10.1201/9781315123905

LAMPIRAN

A. RGB-*Xception*

Epoch 1/50
3/3 - 39s - loss: 1.1888 - accuracy: 0.3958 - val_loss: 0.7740 - val_accuracy: 0.7500
Epoch 2/50
3/3 - 2s - loss: 0.6707 - accuracy: 0.7500 - val_loss: 0.5307 - val_accuracy: 0.7917
Epoch 3/50
3/3 - 2s - loss: 0.5133 - accuracy: 0.8021 - val_loss: 0.5010 - val_accuracy: 0.7083
Epoch 4/50
3/3 - 2s - loss: 0.3606 - accuracy: 0.8646 - val_loss: 0.3731 - val_accuracy: 0.8750
Epoch 5/50
3/3 - 2s - loss: 0.2673 - accuracy: 0.8958 - val_loss: 0.3403 - val_accuracy: 0.8750
Epoch 6/50
3/3 - 2s - loss: 0.2103 - accuracy: 0.9479 - val_loss: 0.2693 - val_accuracy: 0.9167
Epoch 7/50
3/3 - 2s - loss: 0.1928 - accuracy: 0.9479 - val_loss: 0.2638 - val_accuracy: 0.8750
Epoch 8/50
3/3 - 2s - loss: 0.1675 - accuracy: 0.9375 - val_loss: 0.2603 - val_accuracy: 0.9167
Epoch 9/50
3/3 - 2s - loss: 0.1202 - accuracy: 0.9688 - val_loss: 0.2640 - val_accuracy: 0.9167
Epoch 10/50
3/3 - 2s - loss: 0.0864 - accuracy: 1.0000 - val_loss: 0.2119 - val_accuracy: 0.9583
Epoch 11/50
3/3 - 2s - loss: 0.1141 - accuracy: 0.9688 - val_loss: 0.2269 - val_accuracy: 0.9167
Epoch 12/50
3/3 - 2s - loss: 0.0910 - accuracy: 0.9688 - val_loss: 0.2163 - val_accuracy: 0.9167
Epoch 13/50
3/3 - 2s - loss: 0.0684 - accuracy: 0.9896 - val_loss: 0.1985 - val_accuracy: 0.9167
Epoch 14/50
3/3 - 2s - loss: 0.1147 - accuracy: 0.9688 - val_loss: 0.1785 - val_accuracy: 0.9583
Epoch 15/50
3/3 - 2s - loss: 0.0648 - accuracy: 0.9896 - val_loss: 0.2229 - val_accuracy: 0.9583
Epoch 16/50
3/3 - 2s - loss: 0.0662 - accuracy: 0.9896 - val_loss: 0.1982 - val_accuracy: 0.9583
Epoch 17/50
3/3 - 2s - loss: 0.0404 - accuracy: 0.9896 - val_loss: 0.1958 - val_accuracy: 0.9167
Epoch 18/50
3/3 - 2s - loss: 0.0674 - accuracy: 0.9688 - val_loss: 0.2072 - val_accuracy: 0.9167
Epoch 19/50
3/3 - 2s - loss: 0.0284 - accuracy: 1.0000 - val_loss: 0.1924 - val_accuracy: 0.9167
Epoch 20/50
3/3 - 2s - loss: 0.0473 - accuracy: 0.9896 - val_loss: 0.2290 - val_accuracy: 0.9167
Epoch 21/50
3/3 - 2s - loss: 0.0450 - accuracy: 0.9792 - val_loss: 0.2855 - val_accuracy: 0.9167
Epoch 22/50
3/3 - 2s - loss: 0.0335 - accuracy: 1.0000 - val_loss: 0.2590 - val_accuracy: 0.9167
Epoch 23/50
3/3 - 2s - loss: 0.0248 - accuracy: 1.0000 - val_loss: 0.2579 - val_accuracy: 0.9167
Epoch 24/50
3/3 - 2s - loss: 0.0262 - accuracy: 0.9896 - val_loss: 0.2766 - val_accuracy: 0.9167
Epoch 25/50
3/3 - 2s - loss: 0.0200 - accuracy: 1.0000 - val_loss: 0.3228 - val_accuracy: 0.9167
Epoch 26/50
3/3 - 2s - loss: 0.0196 - accuracy: 1.0000 - val_loss: 0.3274 - val_accuracy: 0.9167

```
Epoch 27/50
3/3 - 2s - loss: 0.0247 - accuracy: 1.0000 - val_loss: 0.2679 - val_accuracy: 0.9167
Epoch 28/50
3/3 - 2s - loss: 0.0507 - accuracy: 0.9896 - val_loss: 0.2069 - val_accuracy: 0.9167
Epoch 29/50
3/3 - 2s - loss: 0.0410 - accuracy: 0.9792 - val_loss: 0.2588 - val_accuracy: 0.9167
Epoch 30/50
3/3 - 2s - loss: 0.0324 - accuracy: 0.9896 - val_loss: 0.2941 - val_accuracy: 0.9167
Epoch 31/50
3/3 - 2s - loss: 0.0467 - accuracy: 0.9792 - val_loss: 0.3285 - val_accuracy: 0.9167
Epoch 32/50
3/3 - 2s - loss: 0.0277 - accuracy: 1.0000 - val_loss: 0.2866 - val_accuracy: 0.9167
Epoch 33/50
3/3 - 2s - loss: 0.0216 - accuracy: 1.0000 - val_loss: 0.2795 - val_accuracy: 0.9167
Epoch 34/50
3/3 - 2s - loss: 0.0172 - accuracy: 1.0000 - val_loss: 0.2669 - val_accuracy: 0.9167
Epoch 35/50
3/3 - 2s - loss: 0.0199 - accuracy: 1.0000 - val_loss: 0.2911 - val_accuracy: 0.9167
Epoch 36/50
3/3 - 2s - loss: 0.0186 - accuracy: 1.0000 - val_loss: 0.3539 - val_accuracy: 0.9167
Epoch 37/50
3/3 - 2s - loss: 0.0231 - accuracy: 0.9896 - val_loss: 0.3655 - val_accuracy: 0.9167
Epoch 38/50
3/3 - 2s - loss: 0.0069 - accuracy: 1.0000 - val_loss: 0.3201 - val_accuracy: 0.9167
Epoch 39/50
3/3 - 2s - loss: 0.0103 - accuracy: 1.0000 - val_loss: 0.2929 - val_accuracy: 0.9167
Epoch 40/50
3/3 - 2s - loss: 0.0154 - accuracy: 1.0000 - val_loss: 0.2886 - val_accuracy: 0.9167
Epoch 41/50
3/3 - 2s - loss: 0.0221 - accuracy: 1.0000 - val_loss: 0.2684 - val_accuracy: 0.9167
Epoch 42/50
3/3 - 2s - loss: 0.0158 - accuracy: 1.0000 - val_loss: 0.2786 - val_accuracy: 0.9167
Epoch 43/50
3/3 - 2s - loss: 0.0149 - accuracy: 1.0000 - val_loss: 0.2882 - val_accuracy: 0.9167
Epoch 44/50
3/3 - 2s - loss: 0.0174 - accuracy: 1.0000 - val_loss: 0.3160 - val_accuracy: 0.9167
Epoch 45/50
3/3 - 2s - loss: 0.0120 - accuracy: 1.0000 - val_loss: 0.3911 - val_accuracy: 0.9167
Epoch 46/50
3/3 - 2s - loss: 0.0189 - accuracy: 0.9896 - val_loss: 0.4442 - val_accuracy: 0.9167
Epoch 47/50
3/3 - 2s - loss: 0.0135 - accuracy: 1.0000 - val_loss: 0.4268 - val_accuracy: 0.9167
Epoch 48/50
3/3 - 2s - loss: 0.0060 - accuracy: 1.0000 - val_loss: 0.3965 - val_accuracy: 0.9167
Epoch 49/50
3/3 - 2s - loss: 0.0065 - accuracy: 1.0000 - val_loss: 0.3768 - val_accuracy: 0.9167
Epoch 50/50
3/3 - 2s - loss: 0.0078 - accuracy: 1.0000 - val_loss: 0.3490 - val_accuracy: 0.9167
```

B. RGB-*InceptionV3*

```
Epoch 1/50
3/3 - 44s - loss: 2.0193 - accuracy: 0.3333 - val_loss: 1.0046 - val_accuracy: 0.6667
Epoch 2/50
3/3 - 3s - loss: 1.2482 - accuracy: 0.4479 - val_loss: 0.6994 - val_accuracy: 0.7500
Epoch 3/50
3/3 - 3s - loss: 0.8054 - accuracy: 0.6771 - val_loss: 0.7667 - val_accuracy: 0.6667
Epoch 4/50
3/3 - 3s - loss: 0.7500 - accuracy: 0.6771 - val_loss: 0.4159 - val_accuracy: 0.8750
Epoch 5/50
3/3 - 3s - loss: 0.5041 - accuracy: 0.7396 - val_loss: 0.4427 - val_accuracy: 0.7917
Epoch 6/50
3/3 - 3s - loss: 0.4816 - accuracy: 0.7812 - val_loss: 0.3171 - val_accuracy: 0.9167
Epoch 7/50
3/3 - 3s - loss: 0.2964 - accuracy: 0.8958 - val_loss: 0.4453 - val_accuracy: 0.8333
Epoch 8/50
3/3 - 3s - loss: 0.3529 - accuracy: 0.8542 - val_loss: 0.3692 - val_accuracy: 0.8333
Epoch 9/50
3/3 - 3s - loss: 0.3463 - accuracy: 0.8750 - val_loss: 0.2695 - val_accuracy: 0.9167
Epoch 10/50
3/3 - 3s - loss: 0.1951 - accuracy: 0.9479 - val_loss: 0.3015 - val_accuracy: 0.9167
Epoch 11/50
3/3 - 3s - loss: 0.2212 - accuracy: 0.9271 - val_loss: 0.3660 - val_accuracy: 0.8750
Epoch 12/50
3/3 - 3s - loss: 0.1384 - accuracy: 0.9583 - val_loss: 0.2750 - val_accuracy: 0.8750
Epoch 13/50
3/3 - 3s - loss: 0.1823 - accuracy: 0.9583 - val_loss: 0.2812 - val_accuracy: 0.8750
Epoch 14/50
3/3 - 3s - loss: 0.1626 - accuracy: 0.9375 - val_loss: 0.3034 - val_accuracy: 0.9167
Epoch 15/50
3/3 - 3s - loss: 0.1078 - accuracy: 0.9688 - val_loss: 0.2744 - val_accuracy: 0.8750
Epoch 16/50
3/3 - 3s - loss: 0.1429 - accuracy: 0.9688 - val_loss: 0.2579 - val_accuracy: 0.9167
Epoch 17/50
3/3 - 3s - loss: 0.1343 - accuracy: 0.9271 - val_loss: 0.2703 - val_accuracy: 0.9167
Epoch 18/50
3/3 - 3s - loss: 0.1210 - accuracy: 0.9583 - val_loss: 0.3405 - val_accuracy: 0.9167
Epoch 19/50
3/3 - 3s - loss: 0.1177 - accuracy: 0.9792 - val_loss: 0.2475 - val_accuracy: 0.9167
Epoch 20/50
3/3 - 3s - loss: 0.1066 - accuracy: 0.9792 - val_loss: 0.2305 - val_accuracy: 0.9167
Epoch 21/50
3/3 - 3s - loss: 0.0678 - accuracy: 1.0000 - val_loss: 0.3197 - val_accuracy: 0.9167
Epoch 22/50
3/3 - 3s - loss: 0.1055 - accuracy: 0.9688 - val_loss: 0.2579 - val_accuracy: 0.9167
Epoch 23/50
3/3 - 3s - loss: 0.0471 - accuracy: 1.0000 - val_loss: 0.2308 - val_accuracy: 0.9167
Epoch 24/50
3/3 - 3s - loss: 0.0680 - accuracy: 1.0000 - val_loss: 0.2442 - val_accuracy: 0.8750
Epoch 25/50
3/3 - 3s - loss: 0.0820 - accuracy: 0.9896 - val_loss: 0.3155 - val_accuracy: 0.9167
Epoch 26/50
3/3 - 3s - loss: 0.1298 - accuracy: 0.9583 - val_loss: 0.3022 - val_accuracy: 0.8750
Epoch 27/50
3/3 - 3s - loss: 0.0776 - accuracy: 0.9688 - val_loss: 0.2902 - val_accuracy: 0.9167
Epoch 28/50
```

3/3 - 3s - loss: 0.0744 - accuracy: 0.9583 - val_loss: 0.2864 - val_accuracy: 0.8750
Epoch 29/50
3/3 - 3s - loss: 0.0348 - accuracy: 1.0000 - val_loss: 0.3182 - val_accuracy: 0.9167
Epoch 30/50
3/3 - 3s - loss: 0.0778 - accuracy: 0.9792 - val_loss: 0.3051 - val_accuracy: 0.9167
Epoch 31/50
3/3 - 3s - loss: 0.0459 - accuracy: 1.0000 - val_loss: 0.3033 - val_accuracy: 0.8750
Epoch 32/50
3/3 - 3s - loss: 0.0361 - accuracy: 0.9896 - val_loss: 0.2933 - val_accuracy: 0.8750
Epoch 33/50
3/3 - 3s - loss: 0.1138 - accuracy: 0.9688 - val_loss: 0.3510 - val_accuracy: 0.9167
Epoch 34/50
3/3 - 3s - loss: 0.0369 - accuracy: 0.9896 - val_loss: 0.3579 - val_accuracy: 0.9167
Epoch 35/50
3/3 - 3s - loss: 0.0567 - accuracy: 0.9792 - val_loss: 0.3530 - val_accuracy: 0.9167
Epoch 36/50
3/3 - 3s - loss: 0.0373 - accuracy: 0.9896 - val_loss: 0.3416 - val_accuracy: 0.8750
Epoch 37/50
3/3 - 3s - loss: 0.0361 - accuracy: 0.9896 - val_loss: 0.3530 - val_accuracy: 0.8750
Epoch 38/50
3/3 - 3s - loss: 0.0304 - accuracy: 1.0000 - val_loss: 0.3803 - val_accuracy: 0.8750
Epoch 39/50
3/3 - 3s - loss: 0.0497 - accuracy: 0.9896 - val_loss: 0.3822 - val_accuracy: 0.8750
Epoch 40/50
3/3 - 3s - loss: 0.0382 - accuracy: 1.0000 - val_loss: 0.3774 - val_accuracy: 0.8750
Epoch 41/50
3/3 - 3s - loss: 0.0448 - accuracy: 0.9792 - val_loss: 0.3505 - val_accuracy: 0.8750
Epoch 42/50
3/3 - 3s - loss: 0.0390 - accuracy: 0.9896 - val_loss: 0.3220 - val_accuracy: 0.8750
Epoch 43/50
3/3 - 3s - loss: 0.0263 - accuracy: 1.0000 - val_loss: 0.3112 - val_accuracy: 0.8750
Epoch 44/50
3/3 - 3s - loss: 0.0444 - accuracy: 0.9896 - val_loss: 0.2894 - val_accuracy: 0.8750
Epoch 45/50
3/3 - 3s - loss: 0.0290 - accuracy: 1.0000 - val_loss: 0.2865 - val_accuracy: 0.9167
Epoch 46/50
3/3 - 3s - loss: 0.0395 - accuracy: 0.9896 - val_loss: 0.2993 - val_accuracy: 0.8750
Epoch 47/50
3/3 - 3s - loss: 0.0411 - accuracy: 0.9896 - val_loss: 0.3179 - val_accuracy: 0.8750
Epoch 48/50
3/3 - 3s - loss: 0.0355 - accuracy: 1.0000 - val_loss: 0.3271 - val_accuracy: 0.9167
Epoch 49/50
3/3 - 3s - loss: 0.0504 - accuracy: 0.9896 - val_loss: 0.3167 - val_accuracy: 0.9167
Epoch 50/50
3/3 - 3s - loss: 0.0152 - accuracy: 1.0000 - val_loss: 0.3093 - val_accuracy: 0.8750

C. RGB-*ResNet50*

Epoch 1/50
3/3 - 40s - loss: 2.2125 - accuracy: 0.3229 - val_loss: 1.8027 - val_accuracy: 0.3333
Epoch 2/50
3/3 - 3s - loss: 1.4150 - accuracy: 0.3958 - val_loss: 1.6596 - val_accuracy: 0.3333
Epoch 3/50
3/3 - 3s - loss: 1.6623 - accuracy: 0.2812 - val_loss: 1.2625 - val_accuracy: 0.3333
Epoch 4/50
3/3 - 3s - loss: 1.3849 - accuracy: 0.3438 - val_loss: 1.1904 - val_accuracy: 0.3333
Epoch 5/50
3/3 - 3s - loss: 1.2650 - accuracy: 0.3958 - val_loss: 1.1629 - val_accuracy: 0.3333
Epoch 6/50
3/3 - 3s - loss: 1.3484 - accuracy: 0.2917 - val_loss: 1.1622 - val_accuracy: 0.3333
Epoch 7/50
3/3 - 3s - loss: 1.2428 - accuracy: 0.3958 - val_loss: 1.1957 - val_accuracy: 0.3333
Epoch 8/50
3/3 - 3s - loss: 1.2229 - accuracy: 0.3854 - val_loss: 1.1031 - val_accuracy: 0.3333
Epoch 9/50
3/3 - 3s - loss: 1.2622 - accuracy: 0.2500 - val_loss: 1.1315 - val_accuracy: 0.3333
Epoch 10/50
3/3 - 3s - loss: 1.2180 - accuracy: 0.2917 - val_loss: 1.1399 - val_accuracy: 0.4167
Epoch 11/50
3/3 - 3s - loss: 1.2096 - accuracy: 0.3958 - val_loss: 1.1012 - val_accuracy: 0.3333
Epoch 12/50
3/3 - 3s - loss: 1.1797 - accuracy: 0.3021 - val_loss: 1.1358 - val_accuracy: 0.3333
Epoch 13/50
3/3 - 3s - loss: 1.1203 - accuracy: 0.3438 - val_loss: 1.1142 - val_accuracy: 0.3333
Epoch 14/50
3/3 - 3s - loss: 1.1385 - accuracy: 0.3646 - val_loss: 1.1034 - val_accuracy: 0.3333
Epoch 15/50
3/3 - 3s - loss: 1.1723 - accuracy: 0.3333 - val_loss: 1.1066 - val_accuracy: 0.3333
Epoch 16/50
3/3 - 3s - loss: 1.1298 - accuracy: 0.2812 - val_loss: 1.1179 - val_accuracy: 0.3333
Epoch 17/50
3/3 - 3s - loss: 1.1637 - accuracy: 0.2917 - val_loss: 1.1108 - val_accuracy: 0.2917
Epoch 18/50
3/3 - 3s - loss: 1.1535 - accuracy: 0.2812 - val_loss: 1.1151 - val_accuracy: 0.3333
Epoch 19/50
3/3 - 3s - loss: 1.1306 - accuracy: 0.3542 - val_loss: 1.1089 - val_accuracy: 0.3333
Epoch 20/50
3/3 - 3s - loss: 1.1446 - accuracy: 0.2604 - val_loss: 1.1106 - val_accuracy: 0.2917
Epoch 21/50
3/3 - 3s - loss: 1.1084 - accuracy: 0.4167 - val_loss: 1.1056 - val_accuracy: 0.3750
Epoch 22/50
3/3 - 3s - loss: 1.0912 - accuracy: 0.3958 - val_loss: 1.1072 - val_accuracy: 0.2917
Epoch 23/50
3/3 - 3s - loss: 1.0921 - accuracy: 0.3854 - val_loss: 1.1066 - val_accuracy: 0.2500
Epoch 24/50
3/3 - 3s - loss: 1.0798 - accuracy: 0.4583 - val_loss: 1.1085 - val_accuracy: 0.3333
Epoch 25/50
3/3 - 3s - loss: 1.1177 - accuracy: 0.3229 - val_loss: 1.1052 - val_accuracy: 0.2500
Epoch 26/50
3/3 - 3s - loss: 1.1229 - accuracy: 0.3646 - val_loss: 1.1057 - val_accuracy: 0.3333
Epoch 27/50
3/3 - 3s - loss: 1.1065 - accuracy: 0.3333 - val_loss: 1.1057 - val_accuracy: 0.2500
Epoch 28/50

3/3 - 3s - loss: 1.0796 - accuracy: 0.3854 - val_loss: 1.1086 - val_accuracy: 0.3333
Epoch 29/50
3/3 - 3s - loss: 1.0887 - accuracy: 0.4062 - val_loss: 1.1063 - val_accuracy: 0.3333
Epoch 30/50
3/3 - 3s - loss: 1.1061 - accuracy: 0.3958 - val_loss: 1.1030 - val_accuracy: 0.2083
Epoch 31/50
3/3 - 3s - loss: 1.1099 - accuracy: 0.4375 - val_loss: 1.1065 - val_accuracy: 0.3750
Epoch 32/50
3/3 - 3s - loss: 1.1041 - accuracy: 0.3542 - val_loss: 1.1139 - val_accuracy: 0.3333
Epoch 33/50
3/3 - 3s - loss: 1.1016 - accuracy: 0.3438 - val_loss: 1.1175 - val_accuracy: 0.2917
Epoch 34/50
3/3 - 3s - loss: 1.0822 - accuracy: 0.3438 - val_loss: 1.1156 - val_accuracy: 0.3750
Epoch 35/50
3/3 - 3s - loss: 1.1043 - accuracy: 0.3958 - val_loss: 1.1097 - val_accuracy: 0.3333
Epoch 36/50
3/3 - 3s - loss: 1.0347 - accuracy: 0.5208 - val_loss: 1.1070 - val_accuracy: 0.2917
Epoch 37/50
3/3 - 3s - loss: 1.0693 - accuracy: 0.4688 - val_loss: 1.1115 - val_accuracy: 0.2917
Epoch 38/50
3/3 - 3s - loss: 1.1138 - accuracy: 0.3542 - val_loss: 1.1059 - val_accuracy: 0.2917
Epoch 39/50
3/3 - 3s - loss: 1.1187 - accuracy: 0.3542 - val_loss: 1.1059 - val_accuracy: 0.2500
Epoch 40/50
3/3 - 3s - loss: 1.1053 - accuracy: 0.3333 - val_loss: 1.1063 - val_accuracy: 0.2500
Epoch 41/50
3/3 - 3s - loss: 1.1194 - accuracy: 0.3125 - val_loss: 1.1129 - val_accuracy: 0.3333
Epoch 42/50
3/3 - 3s - loss: 1.1031 - accuracy: 0.4062 - val_loss: 1.1089 - val_accuracy: 0.3333
Epoch 43/50
3/3 - 3s - loss: 1.1017 - accuracy: 0.3333 - val_loss: 1.1099 - val_accuracy: 0.3333
Epoch 44/50
3/3 - 3s - loss: 1.0740 - accuracy: 0.3854 - val_loss: 1.1071 - val_accuracy: 0.3333
Epoch 45/50
3/3 - 3s - loss: 1.0903 - accuracy: 0.3958 - val_loss: 1.1112 - val_accuracy: 0.2500
Epoch 46/50
3/3 - 3s - loss: 1.1252 - accuracy: 0.3229 - val_loss: 1.1099 - val_accuracy: 0.3333
Epoch 47/50
3/3 - 3s - loss: 1.1233 - accuracy: 0.3125 - val_loss: 1.1072 - val_accuracy: 0.3333
Epoch 48/50
3/3 - 3s - loss: 1.0812 - accuracy: 0.3750 - val_loss: 1.1049 - val_accuracy: 0.3333
Epoch 49/50
3/3 - 3s - loss: 1.1004 - accuracy: 0.3750 - val_loss: 1.1095 - val_accuracy: 0.2917
Epoch 50/50
3/3 - 3s - loss: 1.1033 - accuracy: 0.3542 - val_loss: 1.1044 - val_accuracy: 0.2083

D. RGB-*VGG16*

Epoch 1/50
3/3 - 61s - loss: 1.2250 - accuracy: 0.3125 - val_loss: 1.2431 - val_accuracy: 0.3333
Epoch 2/50
3/3 - 3s - loss: 1.2086 - accuracy: 0.3750 - val_loss: 1.1484 - val_accuracy: 0.3750
Epoch 3/50
3/3 - 3s - loss: 1.0743 - accuracy: 0.4167 - val_loss: 1.0087 - val_accuracy: 0.5833
Epoch 4/50
3/3 - 3s - loss: 1.0792 - accuracy: 0.4375 - val_loss: 1.0103 - val_accuracy: 0.5417
Epoch 5/50
3/3 - 3s - loss: 1.0438 - accuracy: 0.5104 - val_loss: 0.9420 - val_accuracy: 0.6250
Epoch 6/50
3/3 - 3s - loss: 0.9447 - accuracy: 0.5417 - val_loss: 0.9462 - val_accuracy: 0.5000
Epoch 7/50
3/3 - 3s - loss: 0.9665 - accuracy: 0.5000 - val_loss: 0.9339 - val_accuracy: 0.5417
Epoch 8/50
3/3 - 3s - loss: 0.9104 - accuracy: 0.6042 - val_loss: 0.8625 - val_accuracy: 0.6667
Epoch 9/50
3/3 - 3s - loss: 0.8803 - accuracy: 0.6458 - val_loss: 0.8389 - val_accuracy: 0.7083
Epoch 10/50
3/3 - 3s - loss: 0.8610 - accuracy: 0.6562 - val_loss: 0.8170 - val_accuracy: 0.7083
Epoch 11/50
3/3 - 3s - loss: 0.8234 - accuracy: 0.7083 - val_loss: 0.8183 - val_accuracy: 0.5417
Epoch 12/50
3/3 - 3s - loss: 0.8057 - accuracy: 0.7083 - val_loss: 0.7933 - val_accuracy: 0.6667
Epoch 13/50
3/3 - 3s - loss: 0.7700 - accuracy: 0.6771 - val_loss: 0.7605 - val_accuracy: 0.7917
Epoch 14/50
3/3 - 3s - loss: 0.7881 - accuracy: 0.6667 - val_loss: 0.7305 - val_accuracy: 0.7083
Epoch 15/50
3/3 - 3s - loss: 0.7528 - accuracy: 0.7396 - val_loss: 0.7320 - val_accuracy: 0.6250
Epoch 16/50
3/3 - 3s - loss: 0.7074 - accuracy: 0.7708 - val_loss: 0.7042 - val_accuracy: 0.7500
Epoch 17/50
3/3 - 3s - loss: 0.7089 - accuracy: 0.7396 - val_loss: 0.6957 - val_accuracy: 0.7917
Epoch 18/50
3/3 - 3s - loss: 0.7156 - accuracy: 0.7396 - val_loss: 0.6749 - val_accuracy: 0.8750
Epoch 19/50
3/3 - 3s - loss: 0.6575 - accuracy: 0.7917 - val_loss: 0.6551 - val_accuracy: 0.8333
Epoch 20/50
3/3 - 3s - loss: 0.6619 - accuracy: 0.7708 - val_loss: 0.6524 - val_accuracy: 0.7083
Epoch 21/50
3/3 - 3s - loss: 0.6125 - accuracy: 0.7708 - val_loss: 0.6269 - val_accuracy: 0.8333
Epoch 22/50
3/3 - 3s - loss: 0.5924 - accuracy: 0.8125 - val_loss: 0.6325 - val_accuracy: 0.8333
Epoch 23/50
3/3 - 3s - loss: 0.6330 - accuracy: 0.7604 - val_loss: 0.6024 - val_accuracy: 0.8333
Epoch 24/50
3/3 - 3s - loss: 0.5678 - accuracy: 0.8229 - val_loss: 0.6017 - val_accuracy: 0.8333
Epoch 25/50
3/3 - 3s - loss: 0.5373 - accuracy: 0.8333 - val_loss: 0.5793 - val_accuracy: 0.8333
Epoch 26/50
3/3 - 3s - loss: 0.5625 - accuracy: 0.8229 - val_loss: 0.5718 - val_accuracy: 0.8333
Epoch 27/50
3/3 - 3s - loss: 0.5364 - accuracy: 0.8229 - val_loss: 0.5907 - val_accuracy: 0.7500
Epoch 28/50

```
3/3 - 3s - loss: 0.5652 - accuracy: 0.8125 - val_loss: 0.5725 - val_accuracy: 0.8333
Epoch 29/50
3/3 - 3s - loss: 0.5269 - accuracy: 0.8438 - val_loss: 0.5451 - val_accuracy: 0.8333
Epoch 30/50
3/3 - 3s - loss: 0.5168 - accuracy: 0.8125 - val_loss: 0.5389 - val_accuracy: 0.8750
Epoch 31/50
3/3 - 3s - loss: 0.4690 - accuracy: 0.8542 - val_loss: 0.5518 - val_accuracy: 0.7917
Epoch 32/50
3/3 - 3s - loss: 0.5116 - accuracy: 0.8229 - val_loss: 0.5357 - val_accuracy: 0.7500
Epoch 33/50
3/3 - 3s - loss: 0.4833 - accuracy: 0.8333 - val_loss: 0.5102 - val_accuracy: 0.8750
Epoch 34/50
3/3 - 3s - loss: 0.4596 - accuracy: 0.8958 - val_loss: 0.5088 - val_accuracy: 0.8750
Epoch 35/50
3/3 - 3s - loss: 0.4440 - accuracy: 0.9271 - val_loss: 0.5098 - val_accuracy: 0.7500
Epoch 36/50
3/3 - 3s - loss: 0.4326 - accuracy: 0.8542 - val_loss: 0.4907 - val_accuracy: 0.8333
Epoch 37/50
3/3 - 3s - loss: 0.4277 - accuracy: 0.8854 - val_loss: 0.4752 - val_accuracy: 0.8750
Epoch 38/50
3/3 - 3s - loss: 0.3885 - accuracy: 0.8854 - val_loss: 0.4848 - val_accuracy: 0.8750
Epoch 39/50
3/3 - 3s - loss: 0.4426 - accuracy: 0.8542 - val_loss: 0.4746 - val_accuracy: 0.8333
Epoch 40/50
3/3 - 3s - loss: 0.4514 - accuracy: 0.8229 - val_loss: 0.4828 - val_accuracy: 0.7917
Epoch 41/50
3/3 - 3s - loss: 0.3524 - accuracy: 0.9167 - val_loss: 0.4592 - val_accuracy: 0.8750
Epoch 42/50
3/3 - 3s - loss: 0.3923 - accuracy: 0.8542 - val_loss: 0.4553 - val_accuracy: 0.8750
Epoch 43/50
3/3 - 3s - loss: 0.3587 - accuracy: 0.9062 - val_loss: 0.4426 - val_accuracy: 0.8333
Epoch 44/50
3/3 - 3s - loss: 0.3590 - accuracy: 0.8958 - val_loss: 0.4335 - val_accuracy: 0.8333
Epoch 45/50
3/3 - 3s - loss: 0.3371 - accuracy: 0.9271 - val_loss: 0.4477 - val_accuracy: 0.8333
Epoch 46/50
3/3 - 3s - loss: 0.3354 - accuracy: 0.8958 - val_loss: 0.4211 - val_accuracy: 0.8750
Epoch 47/50
3/3 - 3s - loss: 0.3577 - accuracy: 0.8750 - val_loss: 0.4074 - val_accuracy: 0.8750
Epoch 48/50
3/3 - 3s - loss: 0.3694 - accuracy: 0.8542 - val_loss: 0.4285 - val_accuracy: 0.7917
Epoch 49/50
3/3 - 3s - loss: 0.3066 - accuracy: 0.9271 - val_loss: 0.4244 - val_accuracy: 0.8333
Epoch 50/50
3/3 - 3s - loss: 0.3144 - accuracy: 0.9062 - val_loss: 0.3980 - val_accuracy: 0.8333
```

E. *Grayscale-Xception*

```
Epoch 1/50
3/3 - 38s - loss: 1.0784 - accuracy: 0.4167 - val_loss: 0.6932 - val_accuracy: 0.6667
Epoch 2/50
3/3 - 2s - loss: 0.6451 - accuracy: 0.6875 - val_loss: 0.4169 - val_accuracy: 0.8750
Epoch 3/50
3/3 - 2s - loss: 0.5144 - accuracy: 0.7917 - val_loss: 0.3064 - val_accuracy: 0.9583
Epoch 4/50
3/3 - 2s - loss: 0.3565 - accuracy: 0.8958 - val_loss: 0.2377 - val_accuracy: 0.9583
Epoch 5/50
3/3 - 2s - loss: 0.3110 - accuracy: 0.9271 - val_loss: 0.2121 - val_accuracy: 0.9583
Epoch 6/50
3/3 - 2s - loss: 0.2503 - accuracy: 0.9167 - val_loss: 0.2201 - val_accuracy: 0.9583
Epoch 7/50
3/3 - 2s - loss: 0.1951 - accuracy: 0.9583 - val_loss: 0.2296 - val_accuracy: 0.9583
Epoch 8/50
3/3 - 2s - loss: 0.1447 - accuracy: 0.9479 - val_loss: 0.1932 - val_accuracy: 0.9583
Epoch 9/50
3/3 - 2s - loss: 0.1721 - accuracy: 0.9583 - val_loss: 0.2323 - val_accuracy: 0.9583
Epoch 10/50
3/3 - 2s - loss: 0.1748 - accuracy: 0.9375 - val_loss: 0.2776 - val_accuracy: 0.9583
Epoch 11/50
3/3 - 2s - loss: 0.1201 - accuracy: 0.9792 - val_loss: 0.2057 - val_accuracy: 0.9583
Epoch 12/50
3/3 - 2s - loss: 0.1077 - accuracy: 0.9688 - val_loss: 0.2015 - val_accuracy: 0.9583
Epoch 13/50
3/3 - 2s - loss: 0.0707 - accuracy: 0.9896 - val_loss: 0.2466 - val_accuracy: 0.9583
Epoch 14/50
3/3 - 2s - loss: 0.0659 - accuracy: 0.9896 - val_loss: 0.2888 - val_accuracy: 0.9167
Epoch 15/50
3/3 - 2s - loss: 0.0736 - accuracy: 0.9896 - val_loss: 0.2692 - val_accuracy: 0.9583
Epoch 16/50
3/3 - 2s - loss: 0.0629 - accuracy: 0.9792 - val_loss: 0.2436 - val_accuracy: 0.9583
Epoch 17/50
3/3 - 2s - loss: 0.0444 - accuracy: 0.9896 - val_loss: 0.2432 - val_accuracy: 0.9583
Epoch 18/50
3/3 - 2s - loss: 0.0760 - accuracy: 0.9688 - val_loss: 0.2824 - val_accuracy: 0.9167
Epoch 19/50
3/3 - 2s - loss: 0.0233 - accuracy: 1.0000 - val_loss: 0.3363 - val_accuracy: 0.9167
Epoch 20/50
3/3 - 2s - loss: 0.0268 - accuracy: 1.0000 - val_loss: 0.3329 - val_accuracy: 0.9167
Epoch 21/50
3/3 - 2s - loss: 0.0313 - accuracy: 0.9896 - val_loss: 0.3269 - val_accuracy: 0.9167
Epoch 22/50
3/3 - 2s - loss: 0.0405 - accuracy: 0.9896 - val_loss: 0.2861 - val_accuracy: 0.9167
Epoch 23/50
3/3 - 2s - loss: 0.0411 - accuracy: 1.0000 - val_loss: 0.2712 - val_accuracy: 0.9583
Epoch 24/50
3/3 - 2s - loss: 0.0364 - accuracy: 0.9896 - val_loss: 0.2872 - val_accuracy: 0.9583
Epoch 25/50
3/3 - 2s - loss: 0.0343 - accuracy: 1.0000 - val_loss: 0.2904 - val_accuracy: 0.9583
Epoch 26/50
3/3 - 2s - loss: 0.0381 - accuracy: 0.9896 - val_loss: 0.2964 - val_accuracy: 0.9583
Epoch 27/50
3/3 - 2s - loss: 0.0210 - accuracy: 1.0000 - val_loss: 0.3287 - val_accuracy: 0.9167
Epoch 28/50
```

3/3 - 2s - loss: 0.0223 - accuracy: 1.0000 - val_loss: 0.3486 - val_accuracy: 0.9167
Epoch 29/50
3/3 - 2s - loss: 0.0306 - accuracy: 0.9896 - val_loss: 0.3567 - val_accuracy: 0.9167
Epoch 30/50
3/3 - 2s - loss: 0.0437 - accuracy: 0.9896 - val_loss: 0.3681 - val_accuracy: 0.9167
Epoch 31/50
3/3 - 2s - loss: 0.0357 - accuracy: 0.9896 - val_loss: 0.3014 - val_accuracy: 0.9167
Epoch 32/50
3/3 - 2s - loss: 0.0325 - accuracy: 1.0000 - val_loss: 0.3039 - val_accuracy: 0.9583
Epoch 33/50
3/3 - 2s - loss: 0.0188 - accuracy: 1.0000 - val_loss: 0.3179 - val_accuracy: 0.9167
Epoch 34/50
3/3 - 2s - loss: 0.0276 - accuracy: 1.0000 - val_loss: 0.3361 - val_accuracy: 0.9167
Epoch 35/50
3/3 - 2s - loss: 0.0149 - accuracy: 1.0000 - val_loss: 0.3521 - val_accuracy: 0.9167
Epoch 36/50
3/3 - 2s - loss: 0.0272 - accuracy: 0.9896 - val_loss: 0.4166 - val_accuracy: 0.9167
Epoch 37/50
3/3 - 2s - loss: 0.0488 - accuracy: 0.9792 - val_loss: 0.4282 - val_accuracy: 0.9167
Epoch 38/50
3/3 - 2s - loss: 0.0153 - accuracy: 1.0000 - val_loss: 0.4708 - val_accuracy: 0.8750
Epoch 39/50
3/3 - 2s - loss: 0.0186 - accuracy: 1.0000 - val_loss: 0.4086 - val_accuracy: 0.9167
Epoch 40/50
3/3 - 2s - loss: 0.0200 - accuracy: 1.0000 - val_loss: 0.3174 - val_accuracy: 0.9167
Epoch 41/50
3/3 - 2s - loss: 0.0280 - accuracy: 1.0000 - val_loss: 0.3662 - val_accuracy: 0.9167
Epoch 42/50
3/3 - 2s - loss: 0.0189 - accuracy: 1.0000 - val_loss: 0.4679 - val_accuracy: 0.9167
Epoch 43/50
3/3 - 2s - loss: 0.0712 - accuracy: 0.9792 - val_loss: 0.3943 - val_accuracy: 0.9167
Epoch 44/50
3/3 - 2s - loss: 0.0183 - accuracy: 1.0000 - val_loss: 0.3290 - val_accuracy: 0.9167
Epoch 45/50
3/3 - 2s - loss: 0.0376 - accuracy: 1.0000 - val_loss: 0.3531 - val_accuracy: 0.9167
Epoch 46/50
3/3 - 2s - loss: 0.0128 - accuracy: 1.0000 - val_loss: 0.4482 - val_accuracy: 0.9167
Epoch 47/50
3/3 - 2s - loss: 0.0227 - accuracy: 1.0000 - val_loss: 0.4280 - val_accuracy: 0.9167
Epoch 48/50
3/3 - 2s - loss: 0.0090 - accuracy: 1.0000 - val_loss: 0.3843 - val_accuracy: 0.9167
Epoch 49/50
3/3 - 2s - loss: 0.0083 - accuracy: 1.0000 - val_loss: 0.3541 - val_accuracy: 0.9167
Epoch 50/50
3/3 - 2s - loss: 0.0054 - accuracy: 1.0000 - val_loss: 0.3393 - val_accuracy: 0.9167

F. *Grayscale-InceptionV3*

```
Epoch 1/50
3/3 - 42s - loss: 1.6429 - accuracy: 0.3333 - val_loss: 1.4501 - val_accuracy: 0.3750
Epoch 2/50
3/3 - 2s - loss: 0.9547 - accuracy: 0.6146 - val_loss: 0.6093 - val_accuracy: 0.7500
Epoch 3/50
3/3 - 2s - loss: 0.7805 - accuracy: 0.6562 - val_loss: 0.4253 - val_accuracy: 0.8333
Epoch 4/50
3/3 - 2s - loss: 0.4296 - accuracy: 0.8542 - val_loss: 0.8137 - val_accuracy: 0.7083
Epoch 5/50
3/3 - 2s - loss: 0.5173 - accuracy: 0.8021 - val_loss: 0.5784 - val_accuracy: 0.7917
Epoch 6/50
3/3 - 2s - loss: 0.3171 - accuracy: 0.8958 - val_loss: 0.4350 - val_accuracy: 0.7083
Epoch 7/50
3/3 - 2s - loss: 0.3351 - accuracy: 0.8750 - val_loss: 0.4934 - val_accuracy: 0.7083
Epoch 8/50
3/3 - 2s - loss: 0.1918 - accuracy: 0.9479 - val_loss: 0.4227 - val_accuracy: 0.8333
Epoch 9/50
3/3 - 2s - loss: 0.2459 - accuracy: 0.8438 - val_loss: 0.6393 - val_accuracy: 0.7500
Epoch 10/50
3/3 - 2s - loss: 0.1916 - accuracy: 0.9479 - val_loss: 0.4231 - val_accuracy: 0.7500
Epoch 11/50
3/3 - 2s - loss: 0.1677 - accuracy: 0.9375 - val_loss: 0.4083 - val_accuracy: 0.8333
Epoch 12/50
3/3 - 2s - loss: 0.1167 - accuracy: 0.9792 - val_loss: 0.4598 - val_accuracy: 0.7500
Epoch 13/50
3/3 - 2s - loss: 0.1741 - accuracy: 0.9479 - val_loss: 0.4996 - val_accuracy: 0.7917
Epoch 14/50
3/3 - 2s - loss: 0.0891 - accuracy: 0.9792 - val_loss: 0.4080 - val_accuracy: 0.7917
Epoch 15/50
3/3 - 2s - loss: 0.1084 - accuracy: 0.9479 - val_loss: 0.4009 - val_accuracy: 0.7917
Epoch 16/50
3/3 - 2s - loss: 0.0845 - accuracy: 0.9896 - val_loss: 0.4743 - val_accuracy: 0.7917
Epoch 17/50
3/3 - 2s - loss: 0.0993 - accuracy: 0.9688 - val_loss: 0.5918 - val_accuracy: 0.7917
Epoch 18/50
3/3 - 2s - loss: 0.0829 - accuracy: 0.9688 - val_loss: 0.5040 - val_accuracy: 0.7917
Epoch 19/50
3/3 - 2s - loss: 0.0724 - accuracy: 0.9792 - val_loss: 0.4455 - val_accuracy: 0.8333
Epoch 20/50
3/3 - 2s - loss: 0.0420 - accuracy: 1.0000 - val_loss: 0.4451 - val_accuracy: 0.7917
Epoch 21/50
3/3 - 2s - loss: 0.1482 - accuracy: 0.9271 - val_loss: 0.5608 - val_accuracy: 0.7917
Epoch 22/50
3/3 - 2s - loss: 0.0533 - accuracy: 1.0000 - val_loss: 0.5264 - val_accuracy: 0.7917
Epoch 23/50
3/3 - 2s - loss: 0.1521 - accuracy: 0.9375 - val_loss: 0.4446 - val_accuracy: 0.8750
Epoch 24/50
3/3 - 2s - loss: 0.0761 - accuracy: 0.9688 - val_loss: 0.4765 - val_accuracy: 0.8333
Epoch 25/50
3/3 - 2s - loss: 0.0436 - accuracy: 1.0000 - val_loss: 0.5139 - val_accuracy: 0.8333
Epoch 26/50
3/3 - 2s - loss: 0.0691 - accuracy: 0.9792 - val_loss: 0.3759 - val_accuracy: 0.8750
Epoch 27/50
3/3 - 2s - loss: 0.0689 - accuracy: 0.9583 - val_loss: 0.4468 - val_accuracy: 0.8750
Epoch 28/50
```

3/3 - 2s - loss: 0.0383 - accuracy: 1.0000 - val_loss: 0.4986 - val_accuracy: 0.8750
Epoch 29/50
3/3 - 2s - loss: 0.0760 - accuracy: 0.9792 - val_loss: 0.4267 - val_accuracy: 0.8750
Epoch 30/50
3/3 - 2s - loss: 0.0443 - accuracy: 0.9896 - val_loss: 0.3829 - val_accuracy: 0.8333
Epoch 31/50
3/3 - 2s - loss: 0.0694 - accuracy: 0.9792 - val_loss: 0.4971 - val_accuracy: 0.8333
Epoch 32/50
3/3 - 2s - loss: 0.0631 - accuracy: 0.9792 - val_loss: 0.5727 - val_accuracy: 0.8333
Epoch 33/50
3/3 - 2s - loss: 0.1008 - accuracy: 0.9792 - val_loss: 0.5765 - val_accuracy: 0.8750
Epoch 34/50
3/3 - 2s - loss: 0.0468 - accuracy: 1.0000 - val_loss: 0.5277 - val_accuracy: 0.8750
Epoch 35/50
3/3 - 2s - loss: 0.0523 - accuracy: 0.9896 - val_loss: 0.4718 - val_accuracy: 0.8750
Epoch 36/50
3/3 - 2s - loss: 0.0477 - accuracy: 0.9896 - val_loss: 0.4788 - val_accuracy: 0.8750
Epoch 37/50
3/3 - 2s - loss: 0.0764 - accuracy: 0.9792 - val_loss: 0.4473 - val_accuracy: 0.8750
Epoch 38/50
3/3 - 2s - loss: 0.0224 - accuracy: 1.0000 - val_loss: 0.4921 - val_accuracy: 0.8750
Epoch 39/50
3/3 - 2s - loss: 0.0529 - accuracy: 0.9896 - val_loss: 0.5314 - val_accuracy: 0.8750
Epoch 40/50
3/3 - 2s - loss: 0.0239 - accuracy: 1.0000 - val_loss: 0.4891 - val_accuracy: 0.8750
Epoch 41/50
3/3 - 2s - loss: 0.0309 - accuracy: 0.9896 - val_loss: 0.4462 - val_accuracy: 0.8750
Epoch 42/50
3/3 - 2s - loss: 0.0255 - accuracy: 1.0000 - val_loss: 0.4716 - val_accuracy: 0.8750
Epoch 43/50
3/3 - 2s - loss: 0.0408 - accuracy: 0.9896 - val_loss: 0.5318 - val_accuracy: 0.8750
Epoch 44/50
3/3 - 2s - loss: 0.0471 - accuracy: 0.9792 - val_loss: 0.5340 - val_accuracy: 0.8750
Epoch 45/50
3/3 - 2s - loss: 0.0304 - accuracy: 0.9896 - val_loss: 0.5760 - val_accuracy: 0.8750
Epoch 46/50
3/3 - 2s - loss: 0.0256 - accuracy: 0.9896 - val_loss: 0.5616 - val_accuracy: 0.8333
Epoch 47/50
3/3 - 2s - loss: 0.0272 - accuracy: 0.9896 - val_loss: 0.5340 - val_accuracy: 0.8750
Epoch 48/50
3/3 - 2s - loss: 0.0200 - accuracy: 1.0000 - val_loss: 0.4842 - val_accuracy: 0.8750
Epoch 49/50
3/3 - 2s - loss: 0.0410 - accuracy: 0.9896 - val_loss: 0.5657 - val_accuracy: 0.8750
Epoch 50/50
3/3 - 2s - loss: 0.0240 - accuracy: 1.0000 - val_loss: 0.6614 - val_accuracy: 0.8333

G. *Grayscale-ResNet50*

```
Epoch 1/50
3/3 - 39s - loss: 2.0803 - accuracy: 0.3333 - val_loss: 1.1556 - val_accuracy: 0.3333
Epoch 2/50
3/3 - 2s - loss: 1.4092 - accuracy: 0.3125 - val_loss: 1.4749 - val_accuracy: 0.3333
Epoch 3/50
3/3 - 2s - loss: 1.4230 - accuracy: 0.3333 - val_loss: 1.2090 - val_accuracy: 0.3333
Epoch 4/50
3/3 - 2s - loss: 1.2729 - accuracy: 0.3333 - val_loss: 1.1930 - val_accuracy: 0.3333
Epoch 5/50
3/3 - 2s - loss: 1.1976 - accuracy: 0.3958 - val_loss: 1.2296 - val_accuracy: 0.3333
Epoch 6/50
3/3 - 2s - loss: 1.3453 - accuracy: 0.3333 - val_loss: 1.1430 - val_accuracy: 0.3333
Epoch 7/50
3/3 - 2s - loss: 1.1901 - accuracy: 0.3438 - val_loss: 1.2564 - val_accuracy: 0.3333
Epoch 8/50
3/3 - 2s - loss: 1.3146 - accuracy: 0.3021 - val_loss: 1.1595 - val_accuracy: 0.3333
Epoch 9/50
3/3 - 2s - loss: 1.2561 - accuracy: 0.2812 - val_loss: 1.1823 - val_accuracy: 0.3333
Epoch 10/50
3/3 - 2s - loss: 1.2492 - accuracy: 0.3021 - val_loss: 1.1226 - val_accuracy: 0.3333
Epoch 11/50
3/3 - 2s - loss: 1.2045 - accuracy: 0.2812 - val_loss: 1.1447 - val_accuracy: 0.3333
Epoch 12/50
3/3 - 2s - loss: 1.1324 - accuracy: 0.3542 - val_loss: 1.1049 - val_accuracy: 0.3333
Epoch 13/50
3/3 - 2s - loss: 1.0864 - accuracy: 0.4062 - val_loss: 1.1322 - val_accuracy: 0.3333
Epoch 14/50
3/3 - 2s - loss: 1.1188 - accuracy: 0.3854 - val_loss: 1.1069 - val_accuracy: 0.4167
Epoch 15/50
3/3 - 2s - loss: 1.1562 - accuracy: 0.2708 - val_loss: 1.1033 - val_accuracy: 0.3333
Epoch 16/50
3/3 - 2s - loss: 1.1123 - accuracy: 0.3333 - val_loss: 1.1050 - val_accuracy: 0.3333
Epoch 17/50
3/3 - 2s - loss: 1.1355 - accuracy: 0.3333 - val_loss: 1.1181 - val_accuracy: 0.2917
Epoch 18/50
3/3 - 2s - loss: 1.1230 - accuracy: 0.3125 - val_loss: 1.1083 - val_accuracy: 0.3333
Epoch 19/50
3/3 - 2s - loss: 1.1600 - accuracy: 0.3021 - val_loss: 1.1037 - val_accuracy: 0.4167
Epoch 20/50
3/3 - 2s - loss: 1.1183 - accuracy: 0.3333 - val_loss: 1.1235 - val_accuracy: 0.3333
Epoch 21/50
3/3 - 2s - loss: 1.1382 - accuracy: 0.3438 - val_loss: 1.1022 - val_accuracy: 0.4167
Epoch 22/50
3/3 - 2s - loss: 1.1166 - accuracy: 0.3542 - val_loss: 1.1058 - val_accuracy: 0.3333
Epoch 23/50
3/3 - 2s - loss: 1.1168 - accuracy: 0.3958 - val_loss: 1.1005 - val_accuracy: 0.3333
Epoch 24/50
3/3 - 2s - loss: 1.1137 - accuracy: 0.4062 - val_loss: 1.1193 - val_accuracy: 0.2917
Epoch 25/50
3/3 - 2s - loss: 1.1152 - accuracy: 0.3750 - val_loss: 1.1004 - val_accuracy: 0.3333
Epoch 26/50
3/3 - 2s - loss: 1.0931 - accuracy: 0.3646 - val_loss: 1.1074 - val_accuracy: 0.3333
Epoch 27/50
3/3 - 2s - loss: 1.1203 - accuracy: 0.2708 - val_loss: 1.1006 - val_accuracy: 0.3750
Epoch 28/50
```

3/3 - 2s - loss: 1.0896 - accuracy: 0.3958 - val_loss: 1.1044 - val_accuracy: 0.3333
Epoch 29/50
3/3 - 2s - loss: 1.1040 - accuracy: 0.3229 - val_loss: 1.1044 - val_accuracy: 0.2083
Epoch 30/50
3/3 - 2s - loss: 1.0984 - accuracy: 0.3333 - val_loss: 1.1017 - val_accuracy: 0.3333
Epoch 31/50
3/3 - 2s - loss: 1.0728 - accuracy: 0.3854 - val_loss: 1.1028 - val_accuracy: 0.3333
Epoch 32/50
3/3 - 2s - loss: 1.0778 - accuracy: 0.4583 - val_loss: 1.0997 - val_accuracy: 0.3333
Epoch 33/50
3/3 - 2s - loss: 1.1068 - accuracy: 0.3333 - val_loss: 1.1071 - val_accuracy: 0.3333
Epoch 34/50
3/3 - 2s - loss: 1.0924 - accuracy: 0.4167 - val_loss: 1.1023 - val_accuracy: 0.4167
Epoch 35/50
3/3 - 2s - loss: 1.0989 - accuracy: 0.3333 - val_loss: 1.1068 - val_accuracy: 0.3333
Epoch 36/50
3/3 - 2s - loss: 1.1263 - accuracy: 0.3229 - val_loss: 1.1091 - val_accuracy: 0.3333
Epoch 37/50
3/3 - 2s - loss: 1.0636 - accuracy: 0.4167 - val_loss: 1.0993 - val_accuracy: 0.3333
Epoch 38/50
3/3 - 2s - loss: 1.1070 - accuracy: 0.3125 - val_loss: 1.1084 - val_accuracy: 0.3333
Epoch 39/50
3/3 - 2s - loss: 1.1088 - accuracy: 0.3542 - val_loss: 1.0981 - val_accuracy: 0.3750
Epoch 40/50
3/3 - 2s - loss: 1.0807 - accuracy: 0.3958 - val_loss: 1.0977 - val_accuracy: 0.3750
Epoch 41/50
3/3 - 2s - loss: 1.1003 - accuracy: 0.3750 - val_loss: 1.0980 - val_accuracy: 0.4167
Epoch 42/50
3/3 - 2s - loss: 1.0815 - accuracy: 0.3958 - val_loss: 1.0998 - val_accuracy: 0.3333
Epoch 43/50
3/3 - 2s - loss: 1.0953 - accuracy: 0.3542 - val_loss: 1.0987 - val_accuracy: 0.3333
Epoch 44/50
3/3 - 2s - loss: 1.1023 - accuracy: 0.3750 - val_loss: 1.1020 - val_accuracy: 0.3333
Epoch 45/50
3/3 - 2s - loss: 1.1012 - accuracy: 0.3229 - val_loss: 1.1014 - val_accuracy: 0.3333
Epoch 46/50
3/3 - 2s - loss: 1.1020 - accuracy: 0.3854 - val_loss: 1.0975 - val_accuracy: 0.3333
Epoch 47/50
3/3 - 2s - loss: 1.0959 - accuracy: 0.3854 - val_loss: 1.0980 - val_accuracy: 0.3750
Epoch 48/50
3/3 - 2s - loss: 1.1177 - accuracy: 0.3021 - val_loss: 1.0984 - val_accuracy: 0.3750
Epoch 49/50
3/3 - 2s - loss: 1.1105 - accuracy: 0.3750 - val_loss: 1.0984 - val_accuracy: 0.3750
Epoch 50/50
3/3 - 2s - loss: 1.0954 - accuracy: 0.3438 - val_loss: 1.0983 - val_accuracy: 0.2917

H. *Grayscale-VGG16*

```
Epoch 1/50
3/3 - 50s - loss: 1.2189 - accuracy: 0.3438 - val_loss: 1.1532 - val_accuracy: 0.3333
Epoch 2/50
3/3 - 2s - loss: 1.2059 - accuracy: 0.3646 - val_loss: 1.0659 - val_accuracy: 0.4167
Epoch 3/50
3/3 - 2s - loss: 1.0718 - accuracy: 0.4688 - val_loss: 0.9883 - val_accuracy: 0.4167
Epoch 4/50
3/3 - 2s - loss: 1.0247 - accuracy: 0.4479 - val_loss: 0.9880 - val_accuracy: 0.4167
Epoch 5/50
3/3 - 2s - loss: 0.9850 - accuracy: 0.5104 - val_loss: 0.9174 - val_accuracy: 0.7500
Epoch 6/50
3/3 - 2s - loss: 0.9609 - accuracy: 0.4792 - val_loss: 0.9077 - val_accuracy: 0.6250
Epoch 7/50
3/3 - 2s - loss: 0.9346 - accuracy: 0.5729 - val_loss: 0.8751 - val_accuracy: 0.6667
Epoch 8/50
3/3 - 2s - loss: 0.9041 - accuracy: 0.6667 - val_loss: 0.8364 - val_accuracy: 0.7083
Epoch 9/50
3/3 - 2s - loss: 0.8361 - accuracy: 0.6979 - val_loss: 0.8364 - val_accuracy: 0.6667
Epoch 10/50
3/3 - 2s - loss: 0.9004 - accuracy: 0.6250 - val_loss: 0.8096 - val_accuracy: 0.6250
Epoch 11/50
3/3 - 2s - loss: 0.8329 - accuracy: 0.6771 - val_loss: 0.7743 - val_accuracy: 0.7500
Epoch 12/50
3/3 - 2s - loss: 0.8085 - accuracy: 0.7083 - val_loss: 0.7585 - val_accuracy: 0.8333
Epoch 13/50
3/3 - 2s - loss: 0.8166 - accuracy: 0.6875 - val_loss: 0.7472 - val_accuracy: 0.7083
Epoch 14/50
3/3 - 2s - loss: 0.7943 - accuracy: 0.7292 - val_loss: 0.7269 - val_accuracy: 0.7500
Epoch 15/50
3/3 - 2s - loss: 0.7453 - accuracy: 0.7500 - val_loss: 0.7126 - val_accuracy: 0.7500
Epoch 16/50
3/3 - 2s - loss: 0.7124 - accuracy: 0.7604 - val_loss: 0.6975 - val_accuracy: 0.7083
Epoch 17/50
3/3 - 2s - loss: 0.7288 - accuracy: 0.7812 - val_loss: 0.6842 - val_accuracy: 0.7083
Epoch 18/50
3/3 - 2s - loss: 0.6616 - accuracy: 0.8021 - val_loss: 0.6727 - val_accuracy: 0.7083
Epoch 19/50
3/3 - 2s - loss: 0.6496 - accuracy: 0.8021 - val_loss: 0.6607 - val_accuracy: 0.7500
Epoch 20/50
3/3 - 2s - loss: 0.6606 - accuracy: 0.8125 - val_loss: 0.6504 - val_accuracy: 0.7917
Epoch 21/50
3/3 - 2s - loss: 0.6971 - accuracy: 0.7708 - val_loss: 0.6506 - val_accuracy: 0.7500
Epoch 22/50
3/3 - 2s - loss: 0.6288 - accuracy: 0.7917 - val_loss: 0.6389 - val_accuracy: 0.7917
Epoch 23/50
3/3 - 2s - loss: 0.6605 - accuracy: 0.7708 - val_loss: 0.6253 - val_accuracy: 0.7083
Epoch 24/50
3/3 - 2s - loss: 0.6015 - accuracy: 0.7917 - val_loss: 0.6238 - val_accuracy: 0.7500
Epoch 25/50
3/3 - 2s - loss: 0.6461 - accuracy: 0.7917 - val_loss: 0.6042 - val_accuracy: 0.7500
Epoch 26/50
3/3 - 2s - loss: 0.6317 - accuracy: 0.7708 - val_loss: 0.5995 - val_accuracy: 0.8333
Epoch 27/50
3/3 - 2s - loss: 0.5704 - accuracy: 0.8229 - val_loss: 0.6043 - val_accuracy: 0.7917
Epoch 28/50
```

3/3 - 2s - loss: 0.5703 - accuracy: 0.8333 - val_loss: 0.6037 - val_accuracy: 0.7500
Epoch 29/50
3/3 - 2s - loss: 0.5652 - accuracy: 0.8542 - val_loss: 0.5845 - val_accuracy: 0.7500
Epoch 30/50
3/3 - 2s - loss: 0.5358 - accuracy: 0.8125 - val_loss: 0.5744 - val_accuracy: 0.7917
Epoch 31/50
3/3 - 2s - loss: 0.4810 - accuracy: 0.8438 - val_loss: 0.5783 - val_accuracy: 0.7917
Epoch 32/50
3/3 - 2s - loss: 0.4908 - accuracy: 0.8854 - val_loss: 0.5746 - val_accuracy: 0.7500
Epoch 33/50
3/3 - 2s - loss: 0.4998 - accuracy: 0.8333 - val_loss: 0.5709 - val_accuracy: 0.7500
Epoch 34/50
3/3 - 2s - loss: 0.4631 - accuracy: 0.8542 - val_loss: 0.5545 - val_accuracy: 0.7500
Epoch 35/50
3/3 - 2s - loss: 0.4854 - accuracy: 0.8750 - val_loss: 0.5479 - val_accuracy: 0.8333
Epoch 36/50
3/3 - 2s - loss: 0.4749 - accuracy: 0.8646 - val_loss: 0.5470 - val_accuracy: 0.7500
Epoch 37/50
3/3 - 2s - loss: 0.4867 - accuracy: 0.8229 - val_loss: 0.5445 - val_accuracy: 0.7500
Epoch 38/50
3/3 - 2s - loss: 0.5186 - accuracy: 0.7917 - val_loss: 0.5301 - val_accuracy: 0.7917
Epoch 39/50
3/3 - 2s - loss: 0.4796 - accuracy: 0.8438 - val_loss: 0.5401 - val_accuracy: 0.7500
Epoch 40/50
3/3 - 2s - loss: 0.4522 - accuracy: 0.8438 - val_loss: 0.5393 - val_accuracy: 0.7500
Epoch 41/50
3/3 - 2s - loss: 0.4387 - accuracy: 0.8854 - val_loss: 0.5300 - val_accuracy: 0.7500
Epoch 42/50
3/3 - 2s - loss: 0.3999 - accuracy: 0.8958 - val_loss: 0.5133 - val_accuracy: 0.7500
Epoch 43/50
3/3 - 2s - loss: 0.4301 - accuracy: 0.8333 - val_loss: 0.5166 - val_accuracy: 0.7917
Epoch 44/50
3/3 - 2s - loss: 0.4204 - accuracy: 0.8542 - val_loss: 0.5328 - val_accuracy: 0.7083
Epoch 45/50
3/3 - 2s - loss: 0.4385 - accuracy: 0.8646 - val_loss: 0.5036 - val_accuracy: 0.7500
Epoch 46/50
3/3 - 2s - loss: 0.4070 - accuracy: 0.8750 - val_loss: 0.5026 - val_accuracy: 0.8333
Epoch 47/50
3/3 - 2s - loss: 0.3900 - accuracy: 0.8958 - val_loss: 0.4995 - val_accuracy: 0.7500
Epoch 48/50
3/3 - 2s - loss: 0.4029 - accuracy: 0.8958 - val_loss: 0.4996 - val_accuracy: 0.7917
Epoch 49/50
3/3 - 2s - loss: 0.3808 - accuracy: 0.9167 - val_loss: 0.5060 - val_accuracy: 0.7500
Epoch 50/50
3/3 - 2s - loss: 0.3604 - accuracy: 0.8854 - val_loss: 0.4955 - val_accuracy: 0.7500

I. HSV-*Xception*

Epoch 1/50
3/3 - 39s - loss: 1.5446 - accuracy: 0.2917 - val_loss: 0.9596 - val_accuracy: 0.5000
Epoch 2/50
3/3 - 2s - loss: 1.0447 - accuracy: 0.4688 - val_loss: 1.0226 - val_accuracy: 0.4583
Epoch 3/50
3/3 - 2s - loss: 0.7585 - accuracy: 0.6667 - val_loss: 0.7115 - val_accuracy: 0.7083
Epoch 4/50
3/3 - 2s - loss: 0.6487 - accuracy: 0.7917 - val_loss: 0.7441 - val_accuracy: 0.6250
Epoch 5/50
3/3 - 2s - loss: 0.5529 - accuracy: 0.8229 - val_loss: 0.6009 - val_accuracy: 0.7917
Epoch 6/50
3/3 - 2s - loss: 0.4487 - accuracy: 0.8438 - val_loss: 0.5924 - val_accuracy: 0.8333
Epoch 7/50
3/3 - 2s - loss: 0.4313 - accuracy: 0.8438 - val_loss: 0.5550 - val_accuracy: 0.8333
Epoch 8/50
3/3 - 2s - loss: 0.2837 - accuracy: 0.9167 - val_loss: 0.5708 - val_accuracy: 0.8333
Epoch 9/50
3/3 - 2s - loss: 0.3395 - accuracy: 0.9062 - val_loss: 0.5206 - val_accuracy: 0.8333
Epoch 10/50
3/3 - 2s - loss: 0.2914 - accuracy: 0.8646 - val_loss: 0.5107 - val_accuracy: 0.7917
Epoch 11/50
3/3 - 2s - loss: 0.2909 - accuracy: 0.8750 - val_loss: 0.5453 - val_accuracy: 0.7917
Epoch 12/50
3/3 - 2s - loss: 0.3774 - accuracy: 0.8750 - val_loss: 0.5247 - val_accuracy: 0.7917
Epoch 13/50
3/3 - 2s - loss: 0.2544 - accuracy: 0.8958 - val_loss: 0.4846 - val_accuracy: 0.8750
Epoch 14/50
3/3 - 2s - loss: 0.2327 - accuracy: 0.8854 - val_loss: 0.5209 - val_accuracy: 0.7917
Epoch 15/50
3/3 - 2s - loss: 0.2743 - accuracy: 0.9062 - val_loss: 0.4696 - val_accuracy: 0.7917
Epoch 16/50
3/3 - 2s - loss: 0.1815 - accuracy: 0.9792 - val_loss: 0.5323 - val_accuracy: 0.8333
Epoch 17/50
3/3 - 2s - loss: 0.2366 - accuracy: 0.8958 - val_loss: 0.5115 - val_accuracy: 0.7917
Epoch 18/50
3/3 - 2s - loss: 0.1967 - accuracy: 0.9375 - val_loss: 0.5384 - val_accuracy: 0.7917
Epoch 19/50
3/3 - 2s - loss: 0.2311 - accuracy: 0.9271 - val_loss: 0.5098 - val_accuracy: 0.8333
Epoch 20/50
3/3 - 2s - loss: 0.1316 - accuracy: 0.9792 - val_loss: 0.5145 - val_accuracy: 0.8333
Epoch 21/50
3/3 - 2s - loss: 0.1113 - accuracy: 0.9792 - val_loss: 0.5045 - val_accuracy: 0.8333
Epoch 22/50
3/3 - 2s - loss: 0.1108 - accuracy: 0.9792 - val_loss: 0.5674 - val_accuracy: 0.7917
Epoch 23/50
3/3 - 2s - loss: 0.1114 - accuracy: 0.9896 - val_loss: 0.5647 - val_accuracy: 0.7917
Epoch 24/50
3/3 - 2s - loss: 0.1003 - accuracy: 0.9896 - val_loss: 0.5592 - val_accuracy: 0.8333
Epoch 25/50
3/3 - 2s - loss: 0.1092 - accuracy: 0.9688 - val_loss: 0.5307 - val_accuracy: 0.8333
Epoch 26/50
3/3 - 2s - loss: 0.0905 - accuracy: 0.9792 - val_loss: 0.5556 - val_accuracy: 0.8333
Epoch 27/50
3/3 - 2s - loss: 0.0821 - accuracy: 1.0000 - val_loss: 0.5658 - val_accuracy: 0.7917
Epoch 28/50

3/3 - 2s - loss: 0.1599 - accuracy: 0.9479 - val_loss: 0.5321 - val_accuracy: 0.7917
Epoch 29/50
3/3 - 2s - loss: 0.0890 - accuracy: 0.9792 - val_loss: 0.6107 - val_accuracy: 0.7917
Epoch 30/50
3/3 - 2s - loss: 0.1516 - accuracy: 0.9375 - val_loss: 0.5018 - val_accuracy: 0.8750
Epoch 31/50
3/3 - 2s - loss: 0.0910 - accuracy: 0.9792 - val_loss: 0.5473 - val_accuracy: 0.8333
Epoch 32/50
3/3 - 2s - loss: 0.1455 - accuracy: 0.9479 - val_loss: 0.4519 - val_accuracy: 0.8750
Epoch 33/50
3/3 - 2s - loss: 0.1120 - accuracy: 0.9479 - val_loss: 0.5350 - val_accuracy: 0.8333
Epoch 34/50
3/3 - 2s - loss: 0.1145 - accuracy: 0.9583 - val_loss: 0.4726 - val_accuracy: 0.7917
Epoch 35/50
3/3 - 2s - loss: 0.0921 - accuracy: 0.9792 - val_loss: 0.5118 - val_accuracy: 0.7917
Epoch 36/50
3/3 - 2s - loss: 0.0877 - accuracy: 0.9896 - val_loss: 0.4059 - val_accuracy: 0.8333
Epoch 37/50
3/3 - 2s - loss: 0.0971 - accuracy: 0.9583 - val_loss: 0.4139 - val_accuracy: 0.8333
Epoch 38/50
3/3 - 2s - loss: 0.0623 - accuracy: 0.9896 - val_loss: 0.4153 - val_accuracy: 0.8333
Epoch 39/50
3/3 - 2s - loss: 0.1052 - accuracy: 0.9583 - val_loss: 0.4346 - val_accuracy: 0.7917
Epoch 40/50
3/3 - 2s - loss: 0.1028 - accuracy: 0.9792 - val_loss: 0.4056 - val_accuracy: 0.8750
Epoch 41/50
3/3 - 2s - loss: 0.0790 - accuracy: 0.9688 - val_loss: 0.4295 - val_accuracy: 0.8750
Epoch 42/50
3/3 - 2s - loss: 0.0524 - accuracy: 0.9896 - val_loss: 0.4498 - val_accuracy: 0.7500
Epoch 43/50
3/3 - 2s - loss: 0.0660 - accuracy: 0.9792 - val_loss: 0.4421 - val_accuracy: 0.7500
Epoch 44/50
3/3 - 2s - loss: 0.0459 - accuracy: 1.0000 - val_loss: 0.4337 - val_accuracy: 0.7917
Epoch 45/50
3/3 - 2s - loss: 0.0426 - accuracy: 0.9896 - val_loss: 0.4397 - val_accuracy: 0.8750
Epoch 46/50
3/3 - 2s - loss: 0.0557 - accuracy: 0.9896 - val_loss: 0.4872 - val_accuracy: 0.7500
Epoch 47/50
3/3 - 2s - loss: 0.0648 - accuracy: 0.9896 - val_loss: 0.5380 - val_accuracy: 0.7500
Epoch 48/50
3/3 - 2s - loss: 0.0742 - accuracy: 0.9688 - val_loss: 0.5256 - val_accuracy: 0.8333
Epoch 49/50
3/3 - 2s - loss: 0.0325 - accuracy: 1.0000 - val_loss: 0.4878 - val_accuracy: 0.8333
Epoch 50/50
3/3 - 2s - loss: 0.0760 - accuracy: 0.9792 - val_loss: 0.4439 - val_accuracy: 0.8333

J. HSV-*InceptionV3*

Epoch 1/50
3/3 - 41s - loss: 1.2562 - accuracy: 0.4375 - val_loss: 0.8002 - val_accuracy: 0.6250
Epoch 2/50
3/3 - 2s - loss: 0.9278 - accuracy: 0.5938 - val_loss: 0.6584 - val_accuracy: 0.6667
Epoch 3/50
3/3 - 2s - loss: 0.6935 - accuracy: 0.7292 - val_loss: 0.9468 - val_accuracy: 0.6250
Epoch 4/50
3/3 - 2s - loss: 0.7330 - accuracy: 0.6354 - val_loss: 0.5315 - val_accuracy: 0.8333
Epoch 5/50
3/3 - 2s - loss: 0.6421 - accuracy: 0.7500 - val_loss: 0.6913 - val_accuracy: 0.7083
Epoch 6/50
3/3 - 2s - loss: 0.4439 - accuracy: 0.8333 - val_loss: 0.6425 - val_accuracy: 0.7500
Epoch 7/50
3/3 - 2s - loss: 0.4895 - accuracy: 0.8229 - val_loss: 0.7019 - val_accuracy: 0.7083
Epoch 8/50
3/3 - 2s - loss: 0.4899 - accuracy: 0.7292 - val_loss: 0.7536 - val_accuracy: 0.7500
Epoch 9/50
3/3 - 2s - loss: 0.3857 - accuracy: 0.8646 - val_loss: 0.5357 - val_accuracy: 0.7500
Epoch 10/50
3/3 - 2s - loss: 0.3214 - accuracy: 0.8854 - val_loss: 0.7219 - val_accuracy: 0.7500
Epoch 11/50
3/3 - 2s - loss: 0.3265 - accuracy: 0.8750 - val_loss: 0.6717 - val_accuracy: 0.7500
Epoch 12/50
3/3 - 2s - loss: 0.2155 - accuracy: 0.9271 - val_loss: 0.5679 - val_accuracy: 0.7500
Epoch 13/50
3/3 - 2s - loss: 0.2869 - accuracy: 0.9062 - val_loss: 0.6114 - val_accuracy: 0.7083
Epoch 14/50
3/3 - 2s - loss: 0.1688 - accuracy: 0.9167 - val_loss: 0.7738 - val_accuracy: 0.7083
Epoch 15/50
3/3 - 2s - loss: 0.1872 - accuracy: 0.9375 - val_loss: 0.5727 - val_accuracy: 0.7917
Epoch 16/50
3/3 - 2s - loss: 0.1889 - accuracy: 0.9479 - val_loss: 0.4845 - val_accuracy: 0.7917
Epoch 17/50
3/3 - 2s - loss: 0.1242 - accuracy: 0.9688 - val_loss: 0.7918 - val_accuracy: 0.7083
Epoch 18/50
3/3 - 2s - loss: 0.2019 - accuracy: 0.9062 - val_loss: 0.5661 - val_accuracy: 0.7500
Epoch 19/50
3/3 - 2s - loss: 0.1836 - accuracy: 0.9375 - val_loss: 0.6081 - val_accuracy: 0.7500
Epoch 20/50
3/3 - 2s - loss: 0.1723 - accuracy: 0.9375 - val_loss: 0.6571 - val_accuracy: 0.7500
Epoch 21/50
3/3 - 2s - loss: 0.2565 - accuracy: 0.8958 - val_loss: 0.6825 - val_accuracy: 0.7917
Epoch 22/50
3/3 - 2s - loss: 0.1636 - accuracy: 0.9375 - val_loss: 0.5083 - val_accuracy: 0.7917
Epoch 23/50
3/3 - 2s - loss: 0.1545 - accuracy: 0.9062 - val_loss: 0.7091 - val_accuracy: 0.7500
Epoch 24/50
3/3 - 2s - loss: 0.1552 - accuracy: 0.9375 - val_loss: 0.8812 - val_accuracy: 0.7083
Epoch 25/50
3/3 - 2s - loss: 0.1368 - accuracy: 0.9479 - val_loss: 0.5661 - val_accuracy: 0.8333
Epoch 26/50
3/3 - 2s - loss: 0.1646 - accuracy: 0.9375 - val_loss: 0.6262 - val_accuracy: 0.7917
Epoch 27/50
3/3 - 2s - loss: 0.0867 - accuracy: 0.9688 - val_loss: 0.8142 - val_accuracy: 0.7500
Epoch 28/50

3/3 - 2s - loss: 0.0812 - accuracy: 0.9792 - val_loss: 0.7146 - val_accuracy: 0.7500
Epoch 29/50
3/3 - 2s - loss: 0.0804 - accuracy: 0.9896 - val_loss: 0.6951 - val_accuracy: 0.7083
Epoch 30/50
3/3 - 2s - loss: 0.0840 - accuracy: 0.9896 - val_loss: 0.6747 - val_accuracy: 0.7083
Epoch 31/50
3/3 - 2s - loss: 0.1101 - accuracy: 0.9479 - val_loss: 0.7411 - val_accuracy: 0.7500
Epoch 32/50
3/3 - 2s - loss: 0.1084 - accuracy: 0.9479 - val_loss: 0.6705 - val_accuracy: 0.7917
Epoch 33/50
3/3 - 2s - loss: 0.1405 - accuracy: 0.9479 - val_loss: 0.5548 - val_accuracy: 0.7917
Epoch 34/50
3/3 - 2s - loss: 0.0817 - accuracy: 0.9792 - val_loss: 0.6049 - val_accuracy: 0.8333
Epoch 35/50
3/3 - 2s - loss: 0.1037 - accuracy: 0.9688 - val_loss: 0.6148 - val_accuracy: 0.7917
Epoch 36/50
3/3 - 2s - loss: 0.1065 - accuracy: 0.9792 - val_loss: 0.5893 - val_accuracy: 0.7917
Epoch 37/50
3/3 - 2s - loss: 0.0778 - accuracy: 0.9896 - val_loss: 0.6745 - val_accuracy: 0.7500
Epoch 38/50
3/3 - 2s - loss: 0.0847 - accuracy: 0.9688 - val_loss: 0.8168 - val_accuracy: 0.7917
Epoch 39/50
3/3 - 2s - loss: 0.1193 - accuracy: 0.9583 - val_loss: 0.6002 - val_accuracy: 0.7500
Epoch 40/50
3/3 - 2s - loss: 0.1459 - accuracy: 0.9583 - val_loss: 0.6378 - val_accuracy: 0.7500
Epoch 41/50
3/3 - 2s - loss: 0.0536 - accuracy: 1.0000 - val_loss: 1.0548 - val_accuracy: 0.7083
Epoch 42/50
3/3 - 2s - loss: 0.0853 - accuracy: 0.9792 - val_loss: 0.8304 - val_accuracy: 0.7083
Epoch 43/50
3/3 - 2s - loss: 0.0639 - accuracy: 0.9896 - val_loss: 0.6490 - val_accuracy: 0.7083
Epoch 44/50
3/3 - 2s - loss: 0.0908 - accuracy: 0.9688 - val_loss: 0.6527 - val_accuracy: 0.7083
Epoch 45/50
3/3 - 2s - loss: 0.0523 - accuracy: 0.9896 - val_loss: 0.8229 - val_accuracy: 0.7500
Epoch 46/50
3/3 - 2s - loss: 0.0988 - accuracy: 0.9583 - val_loss: 0.7874 - val_accuracy: 0.7500
Epoch 47/50
3/3 - 2s - loss: 0.0573 - accuracy: 0.9896 - val_loss: 0.6298 - val_accuracy: 0.7917
Epoch 48/50
3/3 - 2s - loss: 0.0662 - accuracy: 0.9896 - val_loss: 0.5704 - val_accuracy: 0.8333
Epoch 49/50
3/3 - 2s - loss: 0.0368 - accuracy: 1.0000 - val_loss: 0.6415 - val_accuracy: 0.7500
Epoch 50/50
3/3 - 2s - loss: 0.0453 - accuracy: 0.9792 - val_loss: 0.7412 - val_accuracy: 0.7500

K. HSV-*ResNet50*

```
Epoch 1/50
3/3 - 39s - loss: 1.5364 - accuracy: 0.3542 - val_loss: 1.5714 - val_accuracy: 0.3333
Epoch 2/50
3/3 - 2s - loss: 1.7298 - accuracy: 0.2917 - val_loss: 1.2160 - val_accuracy: 0.3333
Epoch 3/50
3/3 - 2s - loss: 1.3716 - accuracy: 0.3646 - val_loss: 1.4994 - val_accuracy: 0.3333
Epoch 4/50
3/3 - 2s - loss: 1.5822 - accuracy: 0.2812 - val_loss: 1.1620 - val_accuracy: 0.3333
Epoch 5/50
3/3 - 2s - loss: 1.3244 - accuracy: 0.3021 - val_loss: 1.2465 - val_accuracy: 0.3333
Epoch 6/50
3/3 - 2s - loss: 1.2397 - accuracy: 0.3438 - val_loss: 1.2022 - val_accuracy: 0.3333
Epoch 7/50
3/3 - 2s - loss: 1.2734 - accuracy: 0.3021 - val_loss: 1.1603 - val_accuracy: 0.2500
Epoch 8/50
3/3 - 2s - loss: 1.2207 - accuracy: 0.2708 - val_loss: 1.2060 - val_accuracy: 0.3333
Epoch 9/50
3/3 - 2s - loss: 1.2504 - accuracy: 0.2812 - val_loss: 1.1005 - val_accuracy: 0.2083
Epoch 10/50
3/3 - 2s - loss: 1.1716 - accuracy: 0.3021 - val_loss: 1.1245 - val_accuracy: 0.3333
Epoch 11/50
3/3 - 2s - loss: 1.1101 - accuracy: 0.3854 - val_loss: 1.1120 - val_accuracy: 0.3333
Epoch 12/50
3/3 - 2s - loss: 1.1244 - accuracy: 0.4167 - val_loss: 1.1150 - val_accuracy: 0.3750
Epoch 13/50
3/3 - 2s - loss: 1.1812 - accuracy: 0.2917 - val_loss: 1.1123 - val_accuracy: 0.3333
Epoch 14/50
3/3 - 2s - loss: 1.1663 - accuracy: 0.3021 - val_loss: 1.0962 - val_accuracy: 0.3750
Epoch 15/50
3/3 - 2s - loss: 1.1432 - accuracy: 0.3542 - val_loss: 1.1271 - val_accuracy: 0.3333
Epoch 16/50
3/3 - 2s - loss: 1.1557 - accuracy: 0.3125 - val_loss: 1.0951 - val_accuracy: 0.3750
Epoch 17/50
3/3 - 2s - loss: 1.1373 - accuracy: 0.3542 - val_loss: 1.1277 - val_accuracy: 0.3333
Epoch 18/50
3/3 - 2s - loss: 1.1554 - accuracy: 0.3125 - val_loss: 1.0978 - val_accuracy: 0.3750
Epoch 19/50
3/3 - 2s - loss: 1.1014 - accuracy: 0.4062 - val_loss: 1.0977 - val_accuracy: 0.4167
Epoch 20/50
3/3 - 2s - loss: 1.1360 - accuracy: 0.3646 - val_loss: 1.0954 - val_accuracy: 0.2917
Epoch 21/50
3/3 - 2s - loss: 1.0918 - accuracy: 0.4062 - val_loss: 1.0990 - val_accuracy: 0.3333
Epoch 22/50
3/3 - 2s - loss: 1.1183 - accuracy: 0.3646 - val_loss: 1.0984 - val_accuracy: 0.2917
Epoch 23/50
3/3 - 2s - loss: 1.1564 - accuracy: 0.2812 - val_loss: 1.0989 - val_accuracy: 0.3750
Epoch 24/50
3/3 - 2s - loss: 1.1327 - accuracy: 0.3125 - val_loss: 1.0944 - val_accuracy: 0.3333
Epoch 25/50
3/3 - 2s - loss: 1.1157 - accuracy: 0.3646 - val_loss: 1.1072 - val_accuracy: 0.3333
Epoch 26/50
3/3 - 2s - loss: 1.1196 - accuracy: 0.3958 - val_loss: 1.1038 - val_accuracy: 0.3333
Epoch 27/50
3/3 - 2s - loss: 1.1344 - accuracy: 0.2917 - val_loss: 1.1221 - val_accuracy: 0.3333
Epoch 28/50
```

3/3 - 2s - loss: 1.1449 - accuracy: 0.3438 - val_loss: 1.0936 - val_accuracy: 0.4167
Epoch 29/50
3/3 - 2s - loss: 1.0824 - accuracy: 0.4167 - val_loss: 1.0977 - val_accuracy: 0.3750
Epoch 30/50
3/3 - 2s - loss: 1.1026 - accuracy: 0.3854 - val_loss: 1.0921 - val_accuracy: 0.4167
Epoch 31/50
3/3 - 2s - loss: 1.1104 - accuracy: 0.3750 - val_loss: 1.0989 - val_accuracy: 0.3750
Epoch 32/50
3/3 - 2s - loss: 1.0959 - accuracy: 0.3646 - val_loss: 1.0911 - val_accuracy: 0.4167
Epoch 33/50
3/3 - 2s - loss: 1.0778 - accuracy: 0.3542 - val_loss: 1.0920 - val_accuracy: 0.3333
Epoch 34/50
3/3 - 2s - loss: 1.0866 - accuracy: 0.3854 - val_loss: 1.0940 - val_accuracy: 0.3750
Epoch 35/50
3/3 - 2s - loss: 1.1218 - accuracy: 0.2812 - val_loss: 1.0955 - val_accuracy: 0.3750
Epoch 36/50
3/3 - 2s - loss: 1.1025 - accuracy: 0.3021 - val_loss: 1.0933 - val_accuracy: 0.3750
Epoch 37/50
3/3 - 2s - loss: 1.1231 - accuracy: 0.3958 - val_loss: 1.0933 - val_accuracy: 0.3333
Epoch 38/50
3/3 - 2s - loss: 1.1014 - accuracy: 0.3333 - val_loss: 1.0942 - val_accuracy: 0.3750
Epoch 39/50
3/3 - 2s - loss: 1.1116 - accuracy: 0.3333 - val_loss: 1.0958 - val_accuracy: 0.3750
Epoch 40/50
3/3 - 2s - loss: 1.0860 - accuracy: 0.4062 - val_loss: 1.0931 - val_accuracy: 0.3750
Epoch 41/50
3/3 - 2s - loss: 1.1123 - accuracy: 0.3958 - val_loss: 1.0948 - val_accuracy: 0.3333
Epoch 42/50
3/3 - 2s - loss: 1.1123 - accuracy: 0.3854 - val_loss: 1.0968 - val_accuracy: 0.3333
Epoch 43/50
3/3 - 2s - loss: 1.0705 - accuracy: 0.3958 - val_loss: 1.0906 - val_accuracy: 0.4167
Epoch 44/50
3/3 - 2s - loss: 1.1066 - accuracy: 0.3438 - val_loss: 1.0982 - val_accuracy: 0.3333
Epoch 45/50
3/3 - 2s - loss: 1.0912 - accuracy: 0.3542 - val_loss: 1.0951 - val_accuracy: 0.3750
Epoch 46/50
3/3 - 2s - loss: 1.0825 - accuracy: 0.4375 - val_loss: 1.0934 - val_accuracy: 0.3750
Epoch 47/50
3/3 - 2s - loss: 1.0985 - accuracy: 0.3438 - val_loss: 1.0926 - val_accuracy: 0.3333
Epoch 48/50
3/3 - 2s - loss: 1.0818 - accuracy: 0.3646 - val_loss: 1.0942 - val_accuracy: 0.3333
Epoch 49/50
3/3 - 2s - loss: 1.0731 - accuracy: 0.3854 - val_loss: 1.0910 - val_accuracy: 0.4167
Epoch 50/50
3/3 - 2s - loss: 1.0846 - accuracy: 0.4167 - val_loss: 1.0933 - val_accuracy: 0.3750

L. HSV-*VGG16*

Epoch 1/50
3/3 - 50s - loss: 1.2090 - accuracy: 0.3229 - val_loss: 1.1358 - val_accuracy: 0.3333
Epoch 2/50
3/3 - 2s - loss: 1.0962 - accuracy: 0.3438 - val_loss: 1.1071 - val_accuracy: 0.3333
Epoch 3/50
3/3 - 2s - loss: 1.1788 - accuracy: 0.3333 - val_loss: 1.0789 - val_accuracy: 0.5000
Epoch 4/50
3/3 - 2s - loss: 1.1157 - accuracy: 0.3646 - val_loss: 1.0707 - val_accuracy: 0.4167
Epoch 5/50
3/3 - 2s - loss: 1.0723 - accuracy: 0.4062 - val_loss: 1.0572 - val_accuracy: 0.3333
Epoch 6/50
3/3 - 2s - loss: 1.0400 - accuracy: 0.4062 - val_loss: 0.9997 - val_accuracy: 0.5000
Epoch 7/50
3/3 - 2s - loss: 0.9825 - accuracy: 0.4583 - val_loss: 0.9916 - val_accuracy: 0.5417
Epoch 8/50
3/3 - 2s - loss: 0.9679 - accuracy: 0.5312 - val_loss: 0.9779 - val_accuracy: 0.5417
Epoch 9/50
3/3 - 2s - loss: 0.9547 - accuracy: 0.5729 - val_loss: 0.9522 - val_accuracy: 0.5417
Epoch 10/50
3/3 - 2s - loss: 0.9294 - accuracy: 0.5312 - val_loss: 0.9487 - val_accuracy: 0.6250
Epoch 11/50
3/3 - 2s - loss: 0.8941 - accuracy: 0.5833 - val_loss: 0.9158 - val_accuracy: 0.5833
Epoch 12/50
3/3 - 2s - loss: 0.8881 - accuracy: 0.5833 - val_loss: 0.9117 - val_accuracy: 0.5833
Epoch 13/50
3/3 - 2s - loss: 0.8784 - accuracy: 0.5729 - val_loss: 0.8897 - val_accuracy: 0.6250
Epoch 14/50
3/3 - 2s - loss: 0.8109 - accuracy: 0.6667 - val_loss: 0.9051 - val_accuracy: 0.6667
Epoch 15/50
3/3 - 2s - loss: 0.8653 - accuracy: 0.6875 - val_loss: 0.8769 - val_accuracy: 0.5417
Epoch 16/50
3/3 - 2s - loss: 0.8124 - accuracy: 0.6771 - val_loss: 0.8614 - val_accuracy: 0.6667
Epoch 17/50
3/3 - 2s - loss: 0.8292 - accuracy: 0.6250 - val_loss: 0.8582 - val_accuracy: 0.6667
Epoch 18/50
3/3 - 2s - loss: 0.7921 - accuracy: 0.6771 - val_loss: 0.8336 - val_accuracy: 0.5417
Epoch 19/50
3/3 - 2s - loss: 0.7731 - accuracy: 0.7083 - val_loss: 0.8307 - val_accuracy: 0.6250
Epoch 20/50
3/3 - 2s - loss: 0.7603 - accuracy: 0.7396 - val_loss: 0.8249 - val_accuracy: 0.5833
Epoch 21/50
3/3 - 2s - loss: 0.7758 - accuracy: 0.6562 - val_loss: 0.8046 - val_accuracy: 0.5833
Epoch 22/50
3/3 - 2s - loss: 0.7719 - accuracy: 0.6667 - val_loss: 0.7979 - val_accuracy: 0.6250
Epoch 23/50
3/3 - 2s - loss: 0.7024 - accuracy: 0.6562 - val_loss: 0.7849 - val_accuracy: 0.5833
Epoch 24/50
3/3 - 2s - loss: 0.7422 - accuracy: 0.6458 - val_loss: 0.7953 - val_accuracy: 0.5833
Epoch 25/50
3/3 - 2s - loss: 0.6860 - accuracy: 0.7292 - val_loss: 0.7736 - val_accuracy: 0.6667
Epoch 26/50
3/3 - 2s - loss: 0.7121 - accuracy: 0.7396 - val_loss: 0.7631 - val_accuracy: 0.5417
Epoch 27/50
3/3 - 2s - loss: 0.6828 - accuracy: 0.7292 - val_loss: 0.7591 - val_accuracy: 0.5833
Epoch 28/50

3/3 - 2s - loss: 0.6337 - accuracy: 0.7396 - val_loss: 0.7557 - val_accuracy: 0.6667
Epoch 29/50
3/3 - 2s - loss: 0.7315 - accuracy: 0.6771 - val_loss: 0.7491 - val_accuracy: 0.5833
Epoch 30/50
3/3 - 2s - loss: 0.6704 - accuracy: 0.6771 - val_loss: 0.7450 - val_accuracy: 0.6250
Epoch 31/50
3/3 - 2s - loss: 0.6591 - accuracy: 0.7083 - val_loss: 0.7366 - val_accuracy: 0.6667
Epoch 32/50
3/3 - 2s - loss: 0.5941 - accuracy: 0.7604 - val_loss: 0.7267 - val_accuracy: 0.6250
Epoch 33/50
3/3 - 2s - loss: 0.6030 - accuracy: 0.7500 - val_loss: 0.7221 - val_accuracy: 0.6250
Epoch 34/50
3/3 - 2s - loss: 0.6100 - accuracy: 0.7917 - val_loss: 0.7248 - val_accuracy: 0.6667
Epoch 35/50
3/3 - 2s - loss: 0.6426 - accuracy: 0.6979 - val_loss: 0.7133 - val_accuracy: 0.6667
Epoch 36/50
3/3 - 2s - loss: 0.6065 - accuracy: 0.7604 - val_loss: 0.7103 - val_accuracy: 0.6250
Epoch 37/50
3/3 - 2s - loss: 0.5995 - accuracy: 0.7500 - val_loss: 0.7040 - val_accuracy: 0.6250
Epoch 38/50
3/3 - 2s - loss: 0.5769 - accuracy: 0.7604 - val_loss: 0.6989 - val_accuracy: 0.6667
Epoch 39/50
3/3 - 2s - loss: 0.6208 - accuracy: 0.7500 - val_loss: 0.7034 - val_accuracy: 0.6667
Epoch 40/50
3/3 - 2s - loss: 0.5243 - accuracy: 0.7917 - val_loss: 0.6893 - val_accuracy: 0.6250
Epoch 41/50
3/3 - 2s - loss: 0.5923 - accuracy: 0.7917 - val_loss: 0.6877 - val_accuracy: 0.6667
Epoch 42/50
3/3 - 2s - loss: 0.5900 - accuracy: 0.7708 - val_loss: 0.6861 - val_accuracy: 0.6667
Epoch 43/50
3/3 - 2s - loss: 0.5548 - accuracy: 0.8229 - val_loss: 0.6759 - val_accuracy: 0.6667
Epoch 44/50
3/3 - 2s - loss: 0.5300 - accuracy: 0.8125 - val_loss: 0.6803 - val_accuracy: 0.6250
Epoch 45/50
3/3 - 2s - loss: 0.5311 - accuracy: 0.8021 - val_loss: 0.6698 - val_accuracy: 0.7083
Epoch 46/50
3/3 - 2s - loss: 0.5195 - accuracy: 0.8125 - val_loss: 0.6635 - val_accuracy: 0.7083
Epoch 47/50
3/3 - 2s - loss: 0.4954 - accuracy: 0.8229 - val_loss: 0.6677 - val_accuracy: 0.6250
Epoch 48/50
3/3 - 2s - loss: 0.4742 - accuracy: 0.8229 - val_loss: 0.6607 - val_accuracy: 0.6667
Epoch 49/50
3/3 - 2s - loss: 0.5208 - accuracy: 0.7708 - val_loss: 0.6754 - val_accuracy: 0.6667
Epoch 50/50
3/3 - 2s - loss: 0.5141 - accuracy: 0.8021 - val_loss: 0.6706 - val_accuracy: 0.6667

A. Double Channel CNN Transfer Learning

```
from google.colab import drive
drive.mount('/content/drive')

import numpy as np
import pickle
import cv2
import seaborn as sn
import pandas as pd
import tensorflow as tf
import matplotlib.pyplot as plt
import os, random, time
import matplotlib.image as mpimg

from keras.applications.xception import Xception
from tensorflow import keras
from keras.preprocessing import image
from keras.layers import Conv2D, MaxPooling2D, GlobalAveragePooling2
D
from keras.layers import Flatten, Dense, Dropout
from keras.layers import BatchNormalization
from os import listdir
from keras.models import Model
from keras import backend as K
from keras.layers import Input
from keras.optimizers import Adam
from keras.callbacks import LearningRateScheduler,ReduceLROnPlateau
from keras.preprocessing.image import ImageDataGenerator
from keras.preprocessing import image
from keras.preprocessing.image import img_to_array
from sklearn.preprocessing import MultiLabelBinarizer
from sklearn.preprocessing import LabelBinarizer
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix, classification_report
from sklearn.datasets import make_classification
from sklearn.preprocessing import label_binarize
from scipy import interp
from itertools import cycle
from sklearn.metrics import roc_curve, auc

EPOCHS = 50
BS = 32
default_image_size = tuple((299, 299))
```

```
image_size = 0
directory_root = "/content/drive/MyDrive/Datasets/rice_leaf_diseases
/"
width=299
height=299
depth=3
labels = os.listdir(directory_root)
print(labels)

def convert_image_to_array(image_dir):
    try:
        image = cv2.imread(image_dir)
        image=cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
        if image is not None :
            image = cv2.resize(image, default_image_size)
            return img_to_array(image)
        else :
            return np.array([])
    except Exception as e:
        print(f"Error : {e}")
        return None

def convert_Enhanced_image_to_array(image_dir):
    try:
        th=10
        max_val=255
        img = cv2.imread(image_dir)
        img_cvt=cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
        img_cvt2=cv2.cvtColor(img_cvt, cv2.COLOR_RGB2GRAY)
        o3 = cv2.cvtColor(img_cvt2, cv2.COLOR_GRAY2RGB)
        if o3 is not None :
            o3 = cv2.resize(o3, default_image_size)
            return img_to_array(o3)
        else :
            return np.array([])
    except Exception as e:
        print(f"Error : {e}")
        return None

def convert_HSV_image_to_array(image_dir):
    try:
        th=10
        max_val=255
        img = cv2.imread(image_dir)
        img_cvt=cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
        o3=cv2.cvtColor(img_cvt, cv2.COLOR_RGB2HSV)
```

```
        if o3 is not None :
            o3 = cv2.resize(o3, default_image_size)
            return img_to_array(o3)
        else :
            return np.array([])
    except Exception as e:
        print(f"Error : {e}")
        return None

# Commented out IPython magic to ensure Python compatibility.
imageEn_list, labelEn_list = [], []
try:
    print("[INFO] Loading images ...")
    root_dir = listdir(directory_root)
    for directory in root_dir :
        # remove .DS_Store from list
        if directory == ".DS_Store" :
            root_dir.remove(directory)

    for folder in root_dir :
        folder_list = listdir(f"{directory_root}/{folder}")

        for image in folder_list[:800]:
            image_directory = f"{directory_root}/{folder}/{image}"
            if image_directory.endswith(".jpg") == True or image_dir
ectory.endswith(".JPG") == True or image_directory.endswith(".jpeg")
 == True or image_directory.endswith(".png") == True:
                imageEn_list.append(convert_Enhanced_image_to_array(
image_directory))
                labelEn_list.append(folder)
#     %time print("[INFO] Image loading completed")
except Exception as e:
    print(f"Error : {e}")

# Commented out IPython magic to ensure Python compatibility.
imageHSV_list, labelHSV_list = [], []
try:
    print("[INFO] Loading images ...")
    root_dir = listdir(directory_root)
    for directory in root_dir :
        # remove .DS_Store from list
        if directory == ".DS_Store" :
            root_dir.remove(directory)

    for folder in root_dir :
        folder_list = listdir(f"{directory_root}/{folder}")
```

```
        for image in folder_list[:800]:
            image_directory = f"{directory_root}/{folder}/{image}"
            if image_directory.endswith(".jpg") == True or image_dir
ectory.endswith(".JPG") == True or image_directory.endswith(".jpeg")
 == True or image_directory.endswith(".png") == True:
                imageHSV_list.append(convert_HSV_image_to_array(imag
e_directory))
                labelHSV_list.append(folder)
#     %time print("[INFO] Image loading completed")
except Exception as e:
    print(f"Error : {e}")

# Commented out IPython magic to ensure Python compatibility.
image_list, label_list = [], []
try:
    print("[INFO] Loading images ...")
    root_dir = listdir(directory_root)
    for directory in root_dir :
        # remove .DS_Store from list
        if directory == ".DS_Store" :
            root_dir.remove(directory)

    for folder in root_dir :
        folder_list = listdir(f"{directory_root}/{folder}")

        for image in folder_list[:800]:
            image_directory = f"{directory_root}/{folder}/{image}"
            if image_directory.endswith(".jpg") == True or image_dir
ectory.endswith(".JPG") == True or image_directory.endswith(".jpeg")
 == True or image_directory.endswith(".png") == True:
                image_list.append(convert_image_to_array(image_direc
tory))
                label_list.append(folder)
#     %time print("[INFO] Image loading completed")
except Exception as e:
    print(f"Error : {e}")

image_size = len(image_list)

print(image_size)

label_binarizer = LabelBinarizer()
image_labels = label_binarizer.fit_transform(label_list)
n_classes = len(label_binarizer.classes_)
labels = os.listdir(directory_root)
```

```
countedlabel = []

# Count total images each classes
file_count = []
for label in labels:
  dir = directory_root + label
  path, dirs, files = next(os.walk(dir))
  file_count.append(len(files))
print("Consist of " + str(len(file_count)) + " Classes")

countedlabel = []
x = 0
while x < len(labels):
  countedlabel.append(labels[x] + " (" + str(file_count[x]) + ")")
  x += 1

plt.figure(figsize=(30,10))
plt.bar(np.arange(len(labels)), file_count, color = ['coral','gold']
)
plt.xticks(np.arange(len(labels)), countedlabel)
plt.title('Dataset Total Image Each Class Information')
plt.xlabel('Total')
plt.ylabel('Labels')
plt.show()

np_image_list = np.array(image_list, dtype=np.float16) / 225.0
np_imageEn_list = np.array(imageEn_list, dtype=np.float16) / 225.0
np_imageHSV_list = np.array(imageHSV_list, dtype=np.float16) / 225.0

x_train, x_test, y_train, y_test = train_test_split(np_image_list, i
mage_labels, test_size=0.2, stratify=image_labels, random_state = 1)
x_Entrain, x_Entest, y_Entrain, y_Entest = train_test_split(np_image
En_list, image_labels, test_size=0.2, stratify=image_labels, random_
state = 1)
x_Hsvtrain, x_Hsvtest, y_Hsvtrain, y_Hsvtest = train_test_split(np_i
mageHSV_list, image_labels, test_size=0.2, stratify=image_labels, ra
ndom_state = 1)

datagen = ImageDataGenerator(
          rotation_range=30,
          width_shift_range=0.2,
          height_shift_range=0.2,
          shear_range=0.15,
          zoom_range=0.15,
          horizontal_flip=True,
          fill_mode="nearest")
```

```
datagen.fit(x_train)
datagen.fit(x_Entrain)
datagen.fit(x_Hsvtrain)

"""**Arsitektur**"""

img_shape = (width, height, depth)

Xception3 = Xception(weights="imagenet", input_shape=img_shape, include_
top=False)
Xception2 = Xception(weights="imagenet", input_shape=img_shape, include_
top=False)
Xception = Xception(weights="imagenet", input_shape=img_shape, include_t
op=False)

def modeld(base_model):
    x = base_model.output
    x = GlobalAveragePooling2D()(x)
    x = Dense(512, activation='relu')(x)
    x = Dropout(0.5)(x)
    model = Model(inputs = base_model.input, outputs = x)
    return model

Xception2._name = "xception_2"
Xception3._name = "xception_3"

for layer in Xception2.layers:
    layer._name = layer._name + str("_2")
    # print(layer._name)

for layer in Xception3.layers:
    layer._name = layer._name + str("_3")
    # print(layer._name)

model1 = modeld(Xception3)
model2 = modeld(Xception)
model3 = modeld(Xception2)

from keras.layers import Flatten, Input, concatenate
combinedOutput = concatenate([model1.output, model2.output])

x = Dense(n_classes, activation="softmax")(combinedOutput)

final_model = Model(inputs=[model1.input, model2.input], outputs=x)
print(final_model.summary())
```

```
from keras.utils.vis_utils import plot_model
plot_model(final_model, to_file='model.png', show_shapes=True, show_
layer_names=True)

# tf.keras.utils.plot_model(
#     final_model,
#     to_file="model.png",
#     show_shapes=True,
#     show_dtype=True,
#     show_layer_names=True,
#     rankdir="TB",
#     expand_nested=True,
#     dpi=100,
# )

Xception3.trainable = False
Xception.trainable = False
Xception2.trainable = False

print(x_train.shape)

final_model.compile(loss=tf.keras.losses.CategoricalCrossentropy(),
                    optimizer=tf.optimizers.Adam(),
                    metrics=['accuracy'])

history = final_model.fit([x_train, x_Entrain], y_train,
                          epochs = EPOCHS, steps_per_epoch=x_train.s
hape[0] // BS,
                          validation_data = ([x_test, x_Entest], y_t
est),
                          verbose=2)

scores = final_model.evaluate([x_test,x_Entest], y_test)
print(f"Test Accuracy: {scores[1]*100}")

acc = history.history['accuracy']
val_acc = history.history['val_accuracy']
loss = history.history['loss']
val_loss = history.history['val_loss']
epochs = range(1, len(acc) + 1)

#Train and validation accuracy
plt.plot(epochs, acc, 'coral', label='Training accurarcy')
plt.plot(epochs, val_acc, 'gold', label='Validation accurarcy')
plt.title('Training and Validation accurarcy')
```

```
plt.legend()

plt.figure()
#Train and validation loss
plt.plot(epochs, loss, 'coral', label='Training loss')
plt.plot(epochs, val_loss, 'gold', label='Validation loss')
plt.title('Training and Validation loss')
plt.legend()
plt.show()

model_pred = final_model.predict([x_test,x_Entest], batch_size=32, v
erbose=2)
model_predicted = np.argmax(model_pred, axis = 1)

#Membuat Confusion Matrix
model_cm = confusion_matrix(np.argmax(y_test, axis=1), model_predict
ed)

#Visualisasi Confusion Matrix
model_df_cm = pd.DataFrame(model_cm, labels, labels)
plt.figure(figsize = (10,7))
sn.set(font_scale=1) #for label size
sn.heatmap(model_df_cm, annot=True, annot_kws={"size": 15}) # font s
ize
plt.show()

model_report = classification_report(np.argmax(y_test, axis=1), mode
l_predicted, target_names=labels )
print(model_report)

# Plot linewidth.
lw = 1

# Compute ROC curve and ROC area for each class
fpr = dict()
tpr = dict()
roc_auc = dict()
for i in range(n_classes):
    fpr[i], tpr[i], _ = roc_curve(y_test[:, i], model_pred[:, i])
    roc_auc[i] = auc(fpr[i], tpr[i])

# Compute micro-average ROC curve and ROC area
fpr["micro"], tpr["micro"], _ = roc_curve(y_test.ravel(), model_pred
.ravel())
roc_auc["micro"] = auc(fpr["micro"], tpr["micro"])
```

```
# Compute macro-average ROC curve and ROC area

# First aggregate all false positive rates
all_fpr = np.unique(np.concatenate([fpr[i] for i in range(n_classes)
]))

# Then interpolate all ROC curves at this points
mean_tpr = np.zeros_like(all_fpr)
for i in range(n_classes):
    mean_tpr += interp(all_fpr, fpr[i], tpr[i])

# Finally average it and compute AUC
mean_tpr /= n_classes

fpr["macro"] = all_fpr
tpr["macro"] = mean_tpr
roc_auc["macro"] = auc(fpr["macro"], tpr["macro"])

# Plot all ROC curves
plt.figure(num=None, figsize=(10, 8), dpi=80, facecolor='w', edgecol
or='k')
plt.plot(fpr["micro"], tpr["micro"],
         label='micro-average ROC curve (area = {0:0.2f})'
               ''.format(roc_auc["micro"]),
         color='deeppink', linestyle=':', linewidth=4)

plt.plot(fpr["macro"], tpr["macro"],
         label='macro-average ROC curve (area = {0:0.2f})'
               ''.format(roc_auc["macro"]),
         color='navy', linestyle=':', linewidth=4)

colors = cycle(['aqua', 'darkorange', 'cornflowerblue'])
for i, color in zip(range(n_classes-97), colors):
    plt.plot(fpr[i], tpr[i], color=color, lw=lw,
             label='ROC curve of class {0} (area = {1:0.2f})'
             ''.format(i, roc_auc[i]))

plt.plot([0, 1], [0, 1], 'k--', lw=lw)
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Some extension of Receiver operating characteristic to mu
lti-class')
plt.legend(loc="lower right")
plt.show()
# Zoom in view of the upper left corner.
```

```
plt.figure(num=None, figsize=(10, 8), dpi=80, facecolor='w', edgecol
or='k')
plt.plot(fpr["micro"], tpr["micro"],
         label='micro-average ROC curve (area = {0:0.2f})'
               ''.format(roc_auc["micro"]),
         color='deeppink', linestyle=':', linewidth=4)

plt.plot(fpr["macro"], tpr["macro"],
         label='macro-average ROC curve (area = {0:0.2f})'
               ''.format(roc_auc["macro"]),
         color='navy', linestyle=':', linewidth=4)

colors = cycle(['aqua', 'darkorange', 'cornflowerblue', 'gray', 'dar
kred', 'forestgreen'])
i=0
for x, color in zip(labels, colors):
    plt.plot(fpr[i], tpr[i], color=color, lw=lw,
             label='ROC curve of class {0} (area = {1:0.2f})'
             ''.format(x, roc_auc[i]))
    i=i+1

plt.plot([0, 1], [0, 1], 'k--', lw=lw)
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Some extension of Receiver operating characteristic to mu
lti-class')
plt.legend(loc="lower right")
plt.show()
```

B. *Three Channel CNN Transfer Learning*

```
from google.colab import drive

drive.mount('/content/drive')

import numpy as np
import pickle
import cv2
import seaborn as sn
import pandas as pd
import tensorflow as tf
import matplotlib.pyplot as plt
import os, random, time
import matplotlib.image as mpimg

from keras.applications.xception import Xception
from tensorflow import keras
from keras.preprocessing import image
from keras.layers import Conv2D, MaxPooling2D, GlobalAveragePooling2
D
from keras.layers import Flatten, Dense, Dropout
from keras.layers import BatchNormalization
from os import listdir
from keras.models import Model
from keras import backend as K
from keras.layers import Input
from keras.optimizers import Adam
from keras.callbacks import LearningRateScheduler,ReduceLROnPlateau
from keras.preprocessing.image import ImageDataGenerator
from keras.preprocessing import image
from keras.preprocessing.image import img_to_array
from sklearn.preprocessing import MultiLabelBinarizer
from sklearn.preprocessing import LabelBinarizer
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix, classification_report
from sklearn.datasets import make_classification
from sklearn.preprocessing import label_binarize
from scipy import interp
from itertools import cycle
from sklearn.metrics import roc_curve, auc

EPOCHS = 50
BS = 32
default_image_size = tuple((299, 299))
image_size = 0
```

```
directory_root = "/content/drive/MyDrive/Datasets/rice_leaf_diseases
/"
width=299
height=299
depth=3
labels = os.listdir(directory_root)
print(labels)

def convert_image_to_array(image_dir):
    try:
        image = cv2.imread(image_dir)
        image=cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
        if image is not None :
            image = cv2.resize(image, default_image_size)
            return img_to_array(image)
        else :
            return np.array([])
    except Exception as e:
        print(f"Error : {e}")
        return None

def convert_Enhanced_image_to_array(image_dir):
    try:
        th=10
        max_val=255
        img = cv2.imread(image_dir)
        img_cvt=cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
        img_cvt2=cv2.cvtColor(img_cvt, cv2.COLOR_RGB2GRAY)
        o3 = cv2.cvtColor(img_cvt2, cv2.COLOR_GRAY2RGB)
        if o3 is not None :
            o3 = cv2.resize(o3, default_image_size)
            return img_to_array(o3)
        else :
            return np.array([])
    except Exception as e:
        print(f"Error : {e}")
        return None

def convert_HSV_image_to_array(image_dir):
    try:
        th=10
        max_val=255
        img = cv2.imread(image_dir)
        img_cvt=cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
        o3=cv2.cvtColor(img_cvt, cv2.COLOR_RGB2HSV)
        if o3 is not None :
```

```
            o3 = cv2.resize(o3, default_image_size)
            return img_to_array(o3)
        else :
            return np.array([])
    except Exception as e:
        print(f"Error : {e}")
        return None

# Commented out IPython magic to ensure Python compatibility.
imageEn_list, labelEn_list = [], []
try:
    print("[INFO] Loading images ...")
    root_dir = listdir(directory_root)
    for directory in root_dir :
        # remove .DS_Store from list
        if directory == ".DS_Store" :
            root_dir.remove(directory)

    for folder in root_dir :
        folder_list = listdir(f"{directory_root}/{folder}")

        for image in folder_list[:800]:
            image_directory = f"{directory_root}/{folder}/{image}"
            if image_directory.endswith(".jpg") == True or image_dir
ectory.endswith(".JPG") == True or image_directory.endswith(".jpeg")
 == True or image_directory.endswith(".png") == True:
                imageEn_list.append(convert_Enhanced_image_to_array(
image_directory))
                labelEn_list.append(folder)
#     %time print("[INFO] Image loading completed")
except Exception as e:
    print(f"Error : {e}")

# Commented out IPython magic to ensure Python compatibility.
imageHSV_list, labelHSV_list = [], []
try:
    print("[INFO] Loading images ...")
    root_dir = listdir(directory_root)
    for directory in root_dir :
        # remove .DS_Store from list
        if directory == ".DS_Store" :
            root_dir.remove(directory)

    for folder in root_dir :
        folder_list = listdir(f"{directory_root}/{folder}")
```

```
        for image in folder_list[:800]:
            image_directory = f"{directory_root}/{folder}/{image}"
            if image_directory.endswith(".jpg") == True or image_dir
ectory.endswith(".JPG") == True or image_directory.endswith(".jpeg")
 == True or image_directory.endswith(".png") == True:
                imageHSV_list.append(convert_HSV_image_to_array(imag
e_directory))
                labelHSV_list.append(folder)
#     %time print("[INFO] Image loading completed")
except Exception as e:
    print(f"Error : {e}")

# Commented out IPython magic to ensure Python compatibility.
image_list, label_list = [], []
try:
    print("[INFO] Loading images ...")
    root_dir = listdir(directory_root)
    for directory in root_dir :
        # remove .DS_Store from list
        if directory == ".DS_Store" :
            root_dir.remove(directory)

    for folder in root_dir :
        folder_list = listdir(f"{directory_root}/{folder}")

        for image in folder_list[:800]:
            image_directory = f"{directory_root}/{folder}/{image}"
            if image_directory.endswith(".jpg") == True or image_dir
ectory.endswith(".JPG") == True or image_directory.endswith(".jpeg")
 == True or image_directory.endswith(".png") == True:
                image_list.append(convert_image_to_array(image_direc
tory))
                label_list.append(folder)
#     %time print("[INFO] Image loading completed")
except Exception as e:
    print(f"Error : {e}")

image_size = len(image_list)

print(image_size)

label_binarizer = LabelBinarizer()
image_labels = label_binarizer.fit_transform(label_list)
n_classes = len(label_binarizer.classes_)
labels = os.listdir(directory_root)
countedlabel = []
```

```
# Count total images each classes
file_count = []
for label in labels:
  dir = directory_root + label
  path, dirs, files = next(os.walk(dir))
  file_count.append(len(files))
print("Consist of " + str(len(file_count)) + " Classes")

countedlabel = []
x = 0
while x < len(labels):
  countedlabel.append(labels[x] + " (" + str(file_count[x]) + ")")
  x += 1

plt.figure(figsize=(30,10))
plt.bar(np.arange(len(labels)), file_count, color = ['coral','gold']
)
plt.xticks(np.arange(len(labels)), countedlabel)
plt.title('Dataset Total Image Each Class Information')
plt.xlabel('Total')
plt.ylabel('Labels')
plt.show()

np_image_list = np.array(image_list, dtype=np.float16) / 225.0
np_imageEn_list = np.array(imageEn_list, dtype=np.float16) / 225.0
np_imageHSV_list = np.array(imageHSV_list, dtype=np.float16) / 225.0

x_train, x_test, y_train, y_test = train_test_split(np_image_list, i
mage_labels, test_size=0.2, stratify=image_labels, random_state = 1)
x_Entrain, x_Entest, y_Entrain, y_Entest = train_test_split(np_image
En_list, image_labels, test_size=0.2, stratify=image_labels, random_
state = 1)
x_Hsvtrain, x_Hsvtest, y_Hsvtrain, y_Hsvtest = train_test_split(np_i
mageHSV_list, image_labels, test_size=0.2, stratify=image_labels, ra
ndom_state = 1)

datagen = ImageDataGenerator(
          rotation_range=30,
          width_shift_range=0.2,
          height_shift_range=0.2,
          shear_range=0.15,
          zoom_range=0.15,
          horizontal_flip=True,
          fill_mode="nearest")
```

```
datagen.fit(x_train)
datagen.fit(x_Entrain)
datagen.fit(x_Hsvtrain)

"""**Arsitektur**"""

img_shape = (width, height, depth)

Xception3 = Xception(weights="imagenet", input_shape=img_shape, incl
ude_top=False)
Xception2 = Xception(weights="imagenet", input_shape=img_shape, incl
ude_top=False)
Xception = Xception(weights="imagenet", input_shape=img_shape, inclu
de_top=False)

Xception3.trainable = False
Xception.trainable = False
Xception2.trainable = False

def modeld(base_model):
    x = base_model.output
    x = GlobalAveragePooling2D()(x)
    x = Dense(512, activation='relu')(x)
    x = Dropout(0.5)(x)
    model = Model(inputs = base_model.input, outputs = x)
    return model

Xception2._name = "xception_2"
Xception3._name = "xception_3"

for layer in Xception2.layers:
    layer._name = layer._name + str("_2")
    # print(layer._name)

for layer in Xception3.layers:
    layer._name = layer._name + str("_3")
    # print(layer._name)

model1 = modeld(Xception3)
model2 = modeld(Xception)
model3 = modeld(Xception2)

from keras.layers import Flatten, Input, concatenate
combinedOutput = concatenate([model1.output, model2.output, model3.o
utput])
```

```
x = Dense(n_classes, activation="softmax")(combinedOutput)

final_model = Model(inputs=[model1.input, model2.input, model3.input
], outputs=x)
print(final_model.summary())

from keras.utils.vis_utils import plot_model
plot_model(final_model, to_file='model.png', show_shapes=True, show_
layer_names=True)

# tf.keras.utils.plot_model(
#     final_model,
#     to_file="model.png",
#     show_shapes=True,
#     show_dtype=True,
#     show_layer_names=True,
#     rankdir="TB",
#     expand_nested=True,
#     dpi=100,
# )

print(x_train.shape)

final_model.compile(loss=tf.keras.losses.CategoricalCrossentropy(),
                    optimizer=tf.optimizers.Adam(),
                    metrics=['accuracy'])

history = final_model.fit([x_train, x_Entrain, x_Hsvtrain], y_train,
                          epochs = EPOCHS, steps_per_epoch=x_train.s
hape[0] // BS,
                          validation_data = ([x_test, x_Entest, x_Hs
vtest], y_test),
                          verbose=2)

scores = final_model.evaluate([x_test,x_Entest,x_Hsvtest], y_test)
print(f"Test Accuracy: {scores[1]*100}")

acc = history.history['accuracy']
val_acc = history.history['val_accuracy']
loss = history.history['loss']
val_loss = history.history['val_loss']
epochs = range(1, len(acc) + 1)

#Train and validation accuracy
plt.plot(epochs, acc, 'coral', label='Training accurarcy')
plt.plot(epochs, val_acc, 'gold', label='Validation accurarcy')
```

```
plt.title('Training and Validation accurarcy')
plt.legend()

plt.figure()
#Train and validation loss
plt.plot(epochs, loss, 'coral', label='Training loss')
plt.plot(epochs, val_loss, 'gold', label='Validation loss')
plt.title('Training and Validation loss')
plt.legend()
plt.show()

model_pred = final_model.predict([x_test,x_Entest, x_Hsvtest], batch
_size=32, verbose=2)
model_predicted = np.argmax(model_pred, axis = 1)

#Membuat Confusion Matrix
model_cm = confusion_matrix(np.argmax(y_test, axis=1), model_predict
ed)

#Visualisasi Confusion Matrix
model_df_cm = pd.DataFrame(model_cm, labels, labels)
plt.figure(figsize = (10,7))
sn.set(font_scale=1) #for label size
sn.heatmap(model_df_cm, annot=True, annot_kws={"size": 15}) # font s
ize
plt.show()

model_report = classification_report(np.argmax(y_test, axis=1), mode
l_predicted, target_names=labels )
print(model_report)

# Plot linewidth.
lw = 1

# Compute ROC curve and ROC area for each class
fpr = dict()
tpr = dict()
roc_auc = dict()
for i in range(n_classes):
    fpr[i], tpr[i], _ = roc_curve(y_test[:, i], model_pred[:, i])
    roc_auc[i] = auc(fpr[i], tpr[i])

# Compute micro-average ROC curve and ROC area
fpr["micro"], tpr["micro"], _ = roc_curve(y_test.ravel(), model_pred
.ravel())
roc_auc["micro"] = auc(fpr["micro"], tpr["micro"])
```

```
# Compute macro-average ROC curve and ROC area

# First aggregate all false positive rates
all_fpr = np.unique(np.concatenate([fpr[i] for i in range(n_classes)
]))

# Then interpolate all ROC curves at this points
mean_tpr = np.zeros_like(all_fpr)
for i in range(n_classes):
    mean_tpr += interp(all_fpr, fpr[i], tpr[i])

# Finally average it and compute AUC
mean_tpr /= n_classes

fpr["macro"] = all_fpr
tpr["macro"] = mean_tpr
roc_auc["macro"] = auc(fpr["macro"], tpr["macro"])

# Plot all ROC curves
plt.figure(num=None, figsize=(10, 8), dpi=80, facecolor='w', edgecol
or='k')
plt.plot(fpr["micro"], tpr["micro"],
         label='micro-average ROC curve (area = {0:0.2f})'
               ''.format(roc_auc["micro"]),
         color='deeppink', linestyle=':', linewidth=4)

plt.plot(fpr["macro"], tpr["macro"],
         label='macro-average ROC curve (area = {0:0.2f})'
               ''.format(roc_auc["macro"]),
         color='navy', linestyle=':', linewidth=4)

colors = cycle(['aqua', 'darkorange', 'cornflowerblue'])
for i, color in zip(range(n_classes-97), colors):
    plt.plot(fpr[i], tpr[i], color=color, lw=lw,
             label='ROC curve of class {0} (area = {1:0.2f})'
             ''.format(i, roc_auc[i]))

plt.plot([0, 1], [0, 1], 'k--', lw=lw)
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Some extension of Receiver operating characteristic to mu
lti-class')
plt.legend(loc="lower right")
plt.show()
```

```
# Zoom in view of the upper left corner.
plt.figure(num=None, figsize=(10, 8), dpi=80, facecolor='w', edgecol
or='k')
plt.plot(fpr["micro"], tpr["micro"],
         label='micro-average ROC curve (area = {0:0.2f})'
               ''.format(roc_auc["micro"]),
         color='deeppink', linestyle=':', linewidth=4)

plt.plot(fpr["macro"], tpr["macro"],
         label='macro-average ROC curve (area = {0:0.2f})'
               ''.format(roc_auc["macro"]),
         color='navy', linestyle=':', linewidth=4)

colors = cycle(['aqua', 'darkorange', 'cornflowerblue', 'gray', 'dar
kred', 'forestgreen'])
i=0
for x, color in zip(labels, colors):
    plt.plot(fpr[i], tpr[i], color=color, lw=lw,
             label='ROC curve of class {0} (area = {1:0.2f})'
             ''.format(x, roc_auc[i]))
    i=i+1

plt.plot([0, 1], [0, 1], 'k--', lw=lw)
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Some extension of Receiver operating characteristic to mu
lti-class')
plt.legend(loc="lower right")
plt.show()
```