CNN Transfer Learning for Visual Guitar Chord Classification

Leon Tran
leontk@stanford.edu

Shawn Zhang
szhang22@stanford.edu

Eric Zhou
ericzhou@stanford.edu

INTRODUCTION

Automatic music chord recognition has always been a challenge, and there has not been much success so far. Previous studies have focused solely on audio; however, we differ by attempting to utilize music's natural visual aspect, starting with visually classifying particular guitar chords. We choose this because guitar chords are linked to unique hand shapes, which should allow us to better classify chords based on visual data.

- This project seeks to build a classifier to determine whether a guitarist is playing a C, D, Em, F, or G chord.
- Our inputs are still images of guitarists with their hands on the fretboard, where we output our predicted classification.

The reason why this is interesting is because music transcription has never yet used visual data, so this could introduce new, improved accuracy. Also, this visual approach could instruct hand posture and finger positions, which is useful for beginners.

DATA

EgoHands Dataset: The EgoHands dataset was used to train MobileNet v2 to extract hands from images. The dataset consists of color videos of people doing various tasks, with pixel-level ground-truth annotations of their hand locations.

Original Dataset: We videotaped five people playing the guitar, collecting over 2 hours of footage. We split these videos into RGB still images and later ran them through MobileNet v2 to bound and crop the hands from the images. These cropped images are our hand-designed features that comprise the dataset.

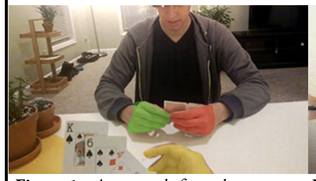

Figure 1a: An example from the EgoHands Dataset.

Figure 1b: An example from our original dataset.

Data Augmentation: To counteract the limitedness and similarity of our original dataset, we applied random horizontal flips and random rotations (-30°, +30°) to help our model generalize. A grayscale copy of the dataset was also generated.

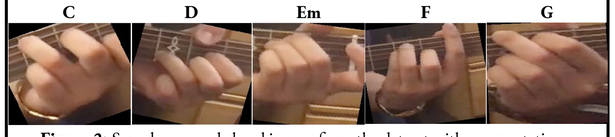

Figure 2: Sample cropped chord images from the dataset with augmentation.

METHODS AND MODELS

Method: Due to our limited data, an end-to-end pipeline was not feasible. Instead, we broke it down into two:

- One pipeline took full images into MobileNet v2 to extract only the fretting hand. These hand images were then resized to 224 x 224, and each color channel was normalized.
- We then fed those images into our second pipeline, a pre-trained neural network for classification.

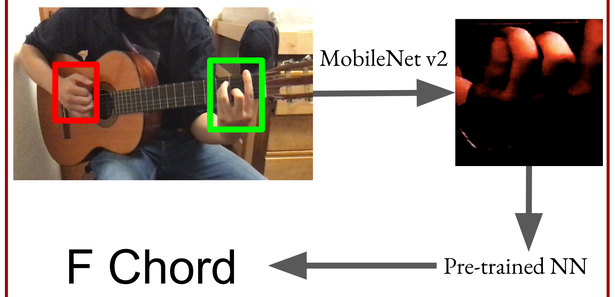

Figure 3: A visual representation of our classification pipeline. The red box indicates the strumming hand, which was ignored. The green box bounds our hand of interest.

Model: Again from our limited data, we were inspired to use transfer learning on the following neural network architectures (which were pre-trained from ImageNet data).

- **ResNet18:**

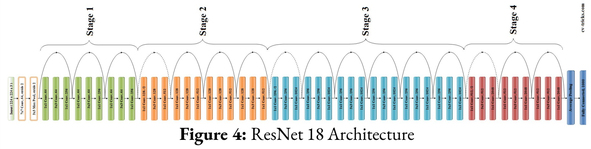

Figure 4: ResNet 18 Architecture

- **GoogLeNet:**

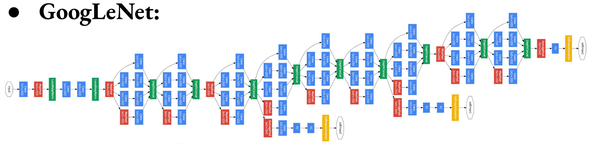

Figure 5: GoogLeNet Architecture

Note, we appended two fully-connected (64-neuron, 32-neuron respectively) layers and a (5-neuron) softmax layer to each network. We also unfroze up to 2 of the last layers in the original network before training on our dataset.

Hyperparameter Search: We experimented with learning rates of 1E-03 and 1E-04, training with or without weight decay, and unfreezing up to 2 layers. Additionally, we also experimented with RGB or grayscale images.

RESULTS

Learning Rate	# Unfrozen Layers	Weight Decay	Train Accuracy	Validation Accuracy	Test Accuracy	Train Loss	Validation Loss	Test Loss
1E-03	0	0	33.6%	51.6%	52.1%	1.61E+00	1.29E+00	1.39E+00
1E-03	1	0	31.3%	47.3%	51.1%	1.60E+00	1.32E+00	1.28E+00
1E-03	2	0	39.1%	50.5%	56.4%	1.42E+00	1.14E+00	1.19E+00
1E-03	0	1e-05	38.9%	48.4%	58.5%	1.48E+00	2.10E+00	1.67E+00
1E-03	1	1e-05	33.7%	46.2%	55.3%	1.52E+00	1.36E+00	1.39E+00
1E-03	2	1e-05	35.8%	54.8%	58.5%	1.46E+00	1.26E+00	1.30E+00
1E-04	0	0	34.9%	49.5%	57.4%	1.45E+00	1.31E+00	1.24E+00
1E-04	1	0	34.2%	34.4%	50.0%	1.46E+00	1.45E+00	1.38E+00
1E-04	2	0	32.9%	39.8%	53.2%	1.45E+00	1.39E+00	1.40E+00
1E-04	0	1e-05	33.6%	57.0%	46.8%	1.45E+00	1.22E+00	1.34E+00
1E-04	1	1e-05	35.6%	44.1%	54.3%	1.45E+00	1.46E+00	1.41E+00
1E-04	2	1e-05	36.2%	41.9%	48.9%	1.45E+00	1.45E+00	1.43E+00

Figure 6: Performance of ResNet18, Greyscale Dataset

Learning Rate	# Unfrozen Layers	Weight Decay	Train Accuracy	Validation Accuracy	Test Accuracy	Train Loss	Validation Loss	Test Loss
1E-03	0	0	53.1%	56.6%	63.1%	1.22E+00	3.28E+00	2.99E+00
1E-03	1	0	64.2%	69.8%	71.3%	1.08E+00	1.40E+00	1.27E+00
1E-03	2	0	87.1%	73.0%	71.9%	6.28E-01	1.30E+00	1.45E+00
1E-03	0	1e-05	30.0%	60.4%	67.5%	1.63E+00	1.07E+00	7.94E-01
1E-03	1	1e-05	79.2%	67.3%	71.9%	7.35E-01	2.34E+00	2.11E+00
1E-03	2	1e-05	89.3%	69.8%	71.9%	4.80E-01	1.96E+00	2.35E+00
1E-04	0	0	33.7%	49.7%	56.9%	1.51E+00	1.20E+00	1.11E+00
1E-04	1	0	36.0%	47.2%	61.3%	1.50E+00	1.36E+00	1.18E+00
1E-04	2	0	36.1%	45.9%	57.5%	1.50E+00	1.24E+00	1.18E+00
1E-04	0	1e-05	33.2%	41.5%	50.0%	1.51E+00	1.45E+00	1.35E+00
1E-04	1	1e-05	35.8%	49.1%	55.6%	1.50E+00	1.30E+00	1.23E+00
1E-04	2	1e-05	33.7%	54.1%	57.5%	1.51E+00	1.18E+00	1.16E+00

Figure 7: Performance of ResNet18, RGB Dataset

Learning Rate	# Unfrozen Layers	Weight Decay	Train Accuracy	Validation Accuracy	Test Accuracy	Train Loss	Validation Loss	Test Loss
1E-03	0	0	100%	98.1%	95.6%	1.75E-04	1.32E-01	1.63E-01
1E-03	1	0	100%	90.6%	95.6%	5.18E-05	2.95E-01	1.74E-01
1E-03	2	0	100%	91.2%	92.5%	1.10E-04	2.15E-01	2.86E-01
1E-03	0	1e-05	100%	84.3%	86.9%	1.53E-03	5.44E-01	4.65E-01
1E-03	1	1e-05	100%	84.9%	88.8%	3.74E-04	5.43E-01	4.34E-01
1E-03	2	1e-05	100%	95.0%	96.3%	2.06E-02	2.29E-01	2.38E-01
1E-04	0	0	100%	98.1%	100.0%	1.12E-04	8.72E-02	5.63E-02
1E-04	1	0	100%	91.2%	95.6%	1.41E-02	3.17E-01	2.12E-01
1E-04	2	0	100%	97.5%	99.4%	2.24E-03	1.84E-01	1.70E-01
1E-04	0	1e-05	100%	98.1%	99.4%	6.22E-03	1.70E-01	1.74E-01
1E-04	1	1e-05	100%	100.0%	98.8%	1.30E-03	1.98E-01	1.99E-01
1E-04	2	1e-05	100%	97.5%	97.5%	7.87E-03	2.36E-01	2.78E-01

Figure 8: Performance of GoogLeNet, RGB Dataset

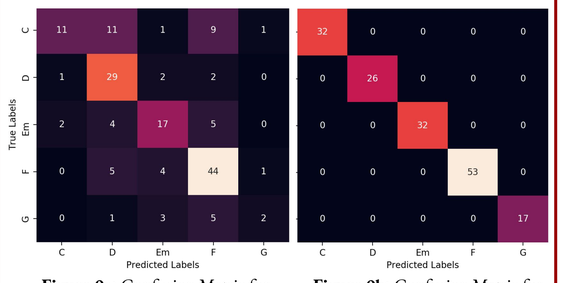

Figure 9a: Confusion Matrix for ResNet18 RGB.

Figure 9b: Confusion Matrix for GoogLeNet RGB.

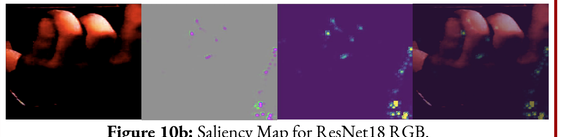

Figure 10b: Saliency Map for ResNet18 RGB.

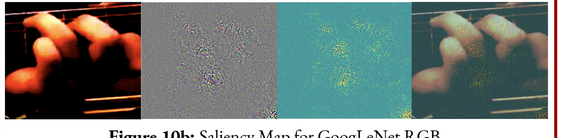

Figure 10b: Saliency Map for GoogLeNet RGB.

DISCUSSION

The models trained on the RGB dataset performed better than those trained on the grayscale dataset.

- Since both neural networks were trained originally on RGB images, this may be due to the loss of transferability or information through contrast in the grayscaling process.

GoogLeNet performed better than ResNet18, even though they were both trained on ImageNet.

- This may be due to the flexibility of the inception architecture and GoogLeNet's lower number of parameters leading to different features downstream.

GoogLeNet yielded high training and test accuracy. Unfortunately, the model did not generalize well to new, "realistic" examples.

- On new examples, the model was mainly accurate on F and Em chords. We believe the poor generalization to be the result of the original dataset not being very diverse - most of the training examples are from similar angles, resolutions, and background lighting conditions.
- This could be fixed by having a larger, more diverse training set or additional methods of image augmentation.

- The saliency map for ResNet18 seems to identify hand outline, while GoogLeNet seems to identify finger and knuckle locations. The ResNet18 confusion matrix indicates that the C chord is most often misclassified. This may be because the finger shape is less unique and resembles those of other chords.

FUTURE WORK

- Make our system work in real-time on video data, allowing faster and more realistic testing.
- Collect more diverse data; hand images with different backgrounds and of different colors, sizes, angles, etc.
- Incorporate audio data into the classification pipeline; more data should yield even better accuracy.
- Play with more architectures trained on different datasets.

REFERENCES

Bambach, S., Lee, S., Crandall, D. J., & Yu, C. (2015). *Lending A Hand: Detecting Hands and Recognizing Activities in Complex Egocentric Interactions.* In ICCV, (pp. 1949–1957).

Misa Ogura (2019, September 26). MisaOgura/flashtorch: 0.1.1 (Version v0.1.1). Zenodo. http://doi.org/10.5281/zenodo.3461737

Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., & Chen, L. C. (2018). MobileNet v2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 4510-4520).