

CPSC 393 Final Project

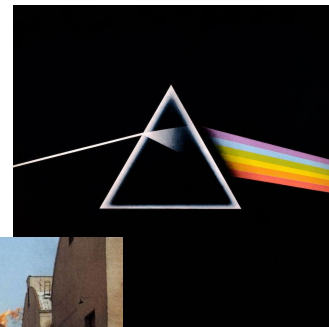
Sreya Vadlamudi, Sarah Fieck



Background

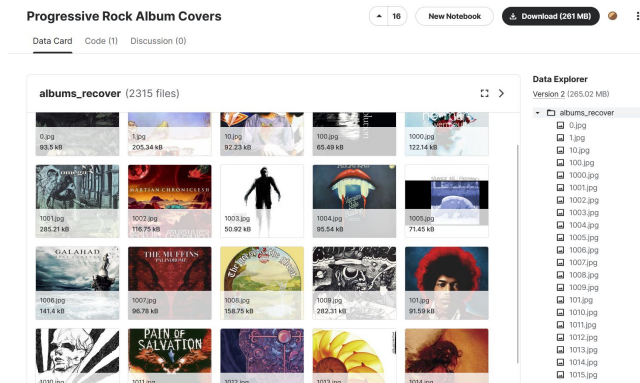
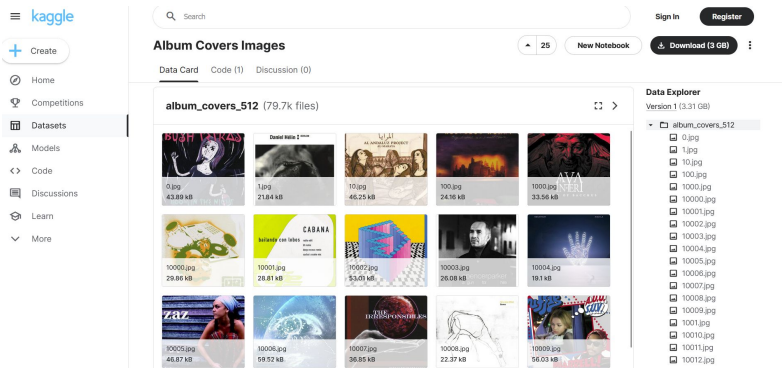
- Progressive Rock (aka Prog Rock) is a subdivision of rock music known for unique musical influences, longer song forms, and conceptual lyrics
- Has unique album cover art, abstract with elements of science fiction, fantasy, and irony

Through this project, we wanted to see how machine learning can detect features of prog album covers in comparison to other album covers.



Data Sets

- Two datasets on Kaggle: One genreless, one prog
- Genreless: 80k of 512 x 512 album covers
- Prog Rock: 2,316 of 260 x 260 album covers
- Did some data cleaning to bring these two datasets together, training & testing file
- Scale the 512 x 512 to 260 x 260



Model & Architecture

- Categorical CNN used to detect features of progressive rock albums when found in the midst of other album covers of different genres
- Model contained seven MaxPooling2D layers for regularization to combat overfitting
- Dropout layers and Batch Normalization were also used for regularization as well
- We did run into an issue where our gradients were all 0 when trying to run our heat map. We fixed this by changing the architecture of our model and added more regularization
- Metrics:
 - Loss: binary_crossentropy
 - Accuracy was measured

Help All changes saved

+ Code + Text

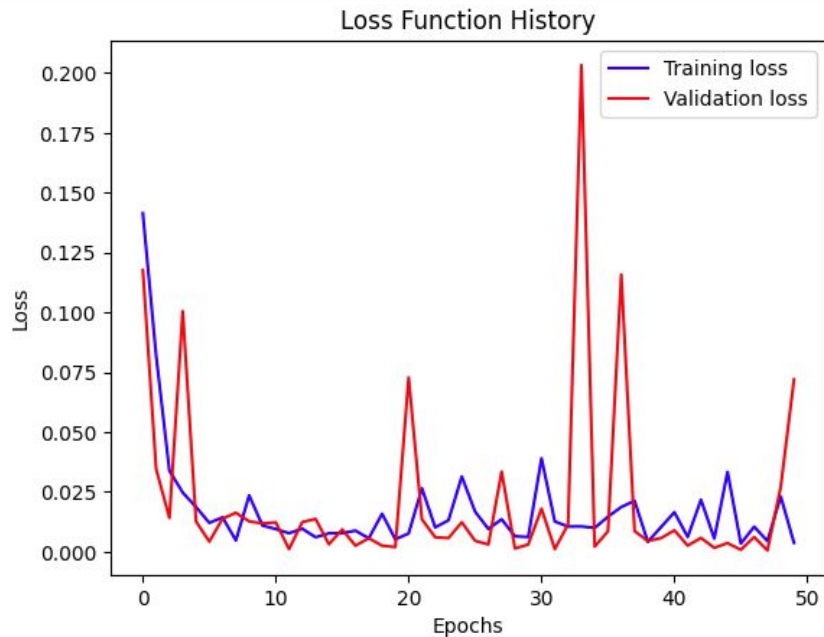
24	✓	●	conv2d (Conv2D)	(None, 258, 258, 32)	896
			max_pooling2d (MaxPooling2D)	(None, 129, 129, 32)	0
			conv2d_1 (Conv2D)	(None, 127, 127, 32)	9248
			max_pooling2d_1 (MaxPooling2D)	(None, 64, 64, 32)	0
			conv2d_2 (Conv2D)	(None, 62, 62, 64)	18496
			max_pooling2d_2 (MaxPooling2D)	(None, 31, 31, 64)	0
			dropout_1 (Dropout)	(None, 31, 31, 64)	0
			conv2d_3 (Conv2D)	(None, 29, 29, 64)	36928
			max_pooling2d_3 (MaxPooling2D)	(None, 15, 15, 64)	0
			conv2d_4 (Conv2D)	(None, 13, 13, 128)	73856
			max_pooling2d_4 (MaxPooling2D)	(None, 7, 7, 128)	0
			dropout_2 (Dropout)	(None, 7, 7, 128)	0
			conv2d_5 (Conv2D)	(None, 5, 5, 128)	147584
			max_pooling2d_5 (MaxPooling2D)	(None, 3, 3, 128)	0
			conv2d_6 (Conv2D)	(None, 1, 1, 128)	147584
			max_pooling2d_6 (MaxPooling2D)	(None, 1, 1, 128)	0
			flatten (Flatten)	(None, 128)	0
			dense (Dense)	(None, 100)	12900
			dense_1 (Dense)	(None, 1)	101

Results

- Ran the model for fifty epochs on the server
- Overall, the model ran smoothly and was not overfit in regards to the loss values and accuracy.
- The model stayed around 99% for accuracy for most of the epochs and the loss .02 by the end.

```
Command Prompt
2023-12-03 18:09:09.390542: I tensorflow/stream_executor/cuda/cuda_dnn.cc:369] Loaded cuDNN version 8201
2052/2052 [=====] - 90s 42ms/step - loss: 0.1414 - accuracy: 0.9713 - val_loss: 0.1177 - val_accuracy: 0.9718
Epoch 2/50
2052/2052 [=====] - 75s 37ms/step - loss: 0.0819 - accuracy: 0.9795 - val_loss: 0.0346 - val_accuracy: 0.9904
Epoch 3/50
2052/2052 [=====] - 74s 36ms/step - loss: 0.0338 - accuracy: 0.9902 - val_loss: 0.0141 - val_accuracy: 0.9940
Epoch 4/50
2052/2052 [=====] - 74s 36ms/step - loss: 0.0248 - accuracy: 0.9935 - val_loss: 0.1005 - val_accuracy: 0.9770
Epoch 5/50
2052/2052 [=====] - 74s 36ms/step - loss: 0.0187 - accuracy: 0.9943 - val_loss: 0.0127 - val_accuracy: 0.9949
Epoch 6/50
2052/2052 [=====] - 73s 36ms/step - loss: 0.0121 - accuracy: 0.9968 - val_loss: 0.0043 - val_accuracy: 0.9987
Epoch 7/50
2052/2052 [=====] - 75s 37ms/step - loss: 0.0144 - accuracy: 0.9965 - val_loss: 0.0139 - val_accuracy: 0.9957
Epoch 8/50
2052/2052 [=====] - 74s 36ms/step - loss: 0.0047 - accuracy: 0.9986 - val_loss: 0.0163 - val_accuracy: 0.9940
Epoch 9/50
2052/2052 [=====] - 72s 35ms/step - loss: 0.0235 - accuracy: 0.9941 - val_loss: 0.0127 - val_accuracy: 0.9958
Epoch 10/50
345/2052 [====>.....] - ETA: 52s - loss: 0.0072 - accuracy: 0.9974
2052/2052 [=====] - 72s 35ms/step - loss: 0.0110 - accuracy: 0.9972 - val_loss: 0.0119 - val_accuracy: 0.9958
Epoch 11/50
2052/2052 [=====] - 73s 35ms/step - loss: 0.0095 - accuracy: 0.9974 - val_loss: 0.0123 - val_accuracy: 0.9961
Epoch 12/50
2052/2052 [=====] - 74s 36ms/step - loss: 0.0078 - accuracy: 0.9980 - val_loss: 0.0011 - val_accuracy: 0.9998
Epoch 13/50
2052/2052 [=====] - 73s 35ms/step - loss: 0.0096 - accuracy: 0.9977 - val_loss: 0.0124 - val_accuracy: 0.9961
Epoch 14/50
2052/2052 [=====] - 73s 36ms/step - loss: 0.0061 - accuracy: 0.9985 - val_loss: 0.0137 - val_accuracy: 0.9957
Epoch 15/50
2052/2052 [=====] - 72s 35ms/step - loss: 0.0077 - accuracy: 0.9980 - val_loss: 0.0032 - val_accuracy: 0.9995
Epoch 16/50
2052/2052 [=====] - 73s 36ms/step - loss: 0.0078 - accuracy: 0.9984 - val_loss: 0.0093 - val_accuracy: 0.9973
Epoch 17/50
2052/2052 [=====] - 72s 35ms/step - loss: 0.0089 - accuracy: 0.9986 - val_loss: 0.0026 - val_accuracy: 0.9993
Epoch 18/50
2052/2052 [=====] - 73s 35ms/step - loss: 0.0056 - accuracy: 0.9985 - val_loss: 0.0057 - val_accuracy: 0.9975
Epoch 19/50
2052/2052 [=====] - 72s 35ms/step - loss: 0.0159 - accuracy: 0.9962 - val_loss: 0.0026 - val_accuracy: 0.9991
```

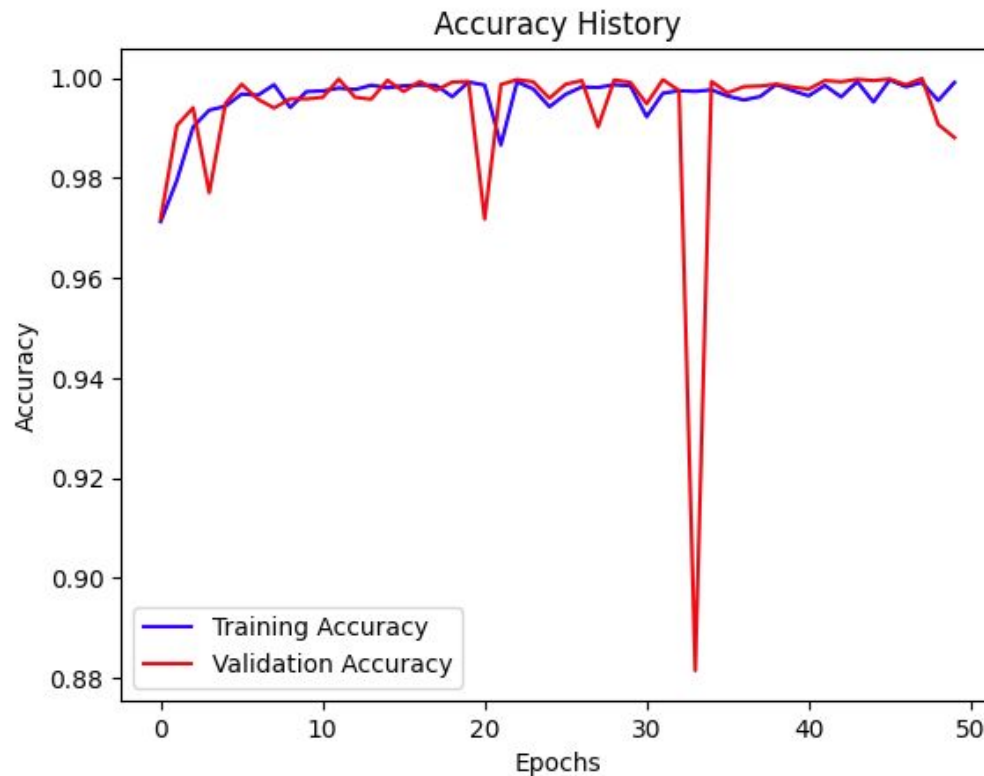
Loss Function Metrics



- Used binary cross-entropy
- Training loss was fairly low, sticking around 0.02 towards later epochs
- Validation loss mostly mimicked this trend, with some abnormalities
 - The spikes?
 - Does not fully exhibit overfitting

Accuracy Metrics

- Training accuracy is highly accurate, above 0.99
- Validation accuracy hugs most of training accuracy, with some dips



Heatmap & Layer Activation

- We created a heatmap in order to see which features of the album covers were most important when comparing prog rock to other albums.
- We compared an image from the prog rock album to an image in the other album.
 - These are the two images(prog rock being on the left):



Image 1:
Album is
Images and
Words by a
progressive
metal band,
Dream
Theater.

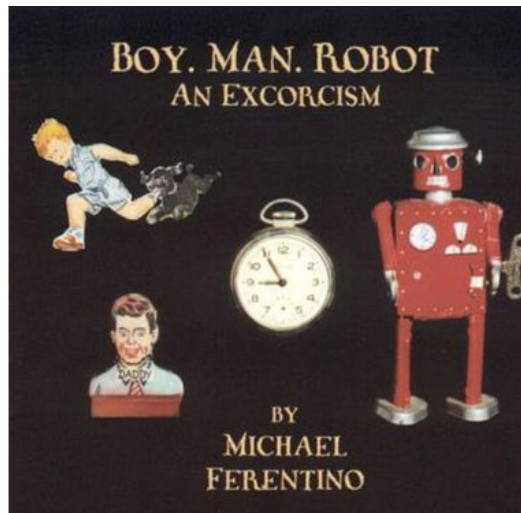
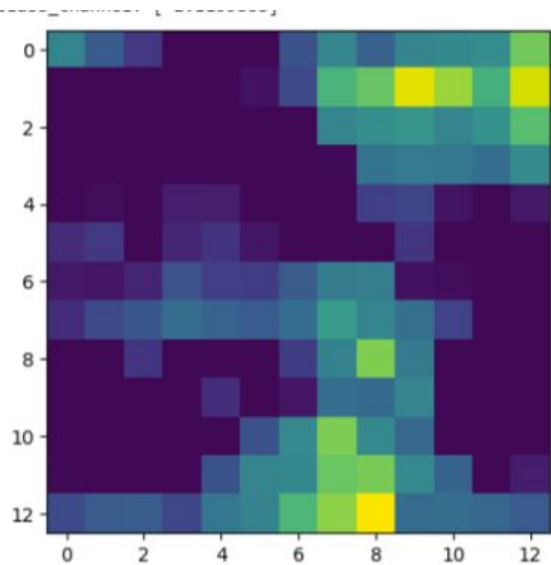


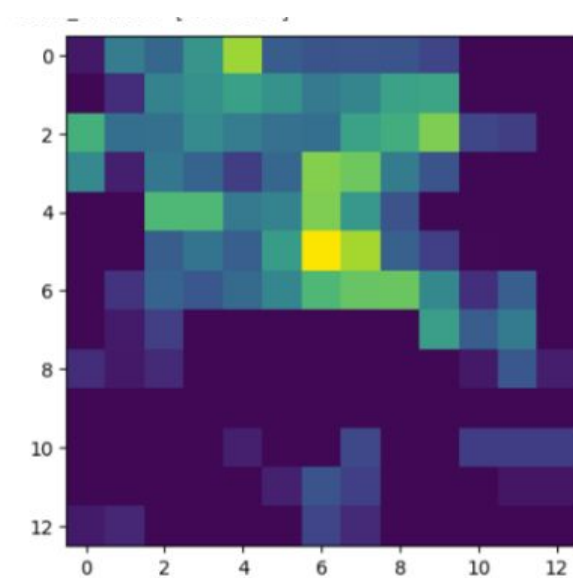
Image 2:
Album is Boy.
Man. Robot. An
Exorcism by
Michael
Ferentino within
the Genreless
album dataset.

Heatmaps for the Two Images

Prog Rock Heatmap



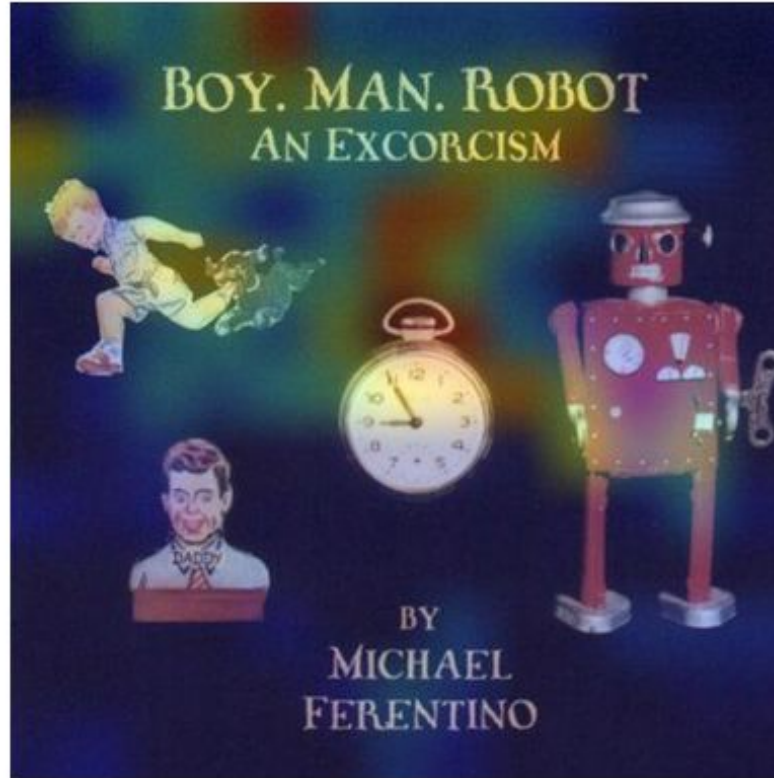
Genreless Heatmap



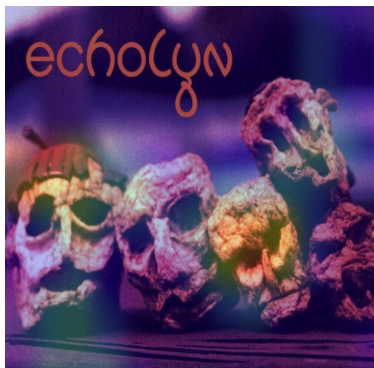
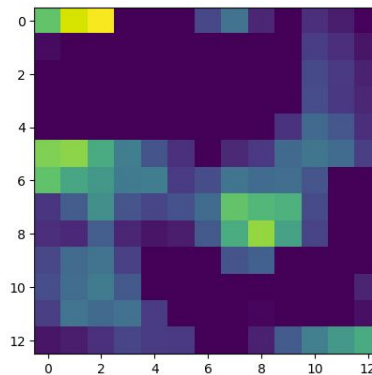
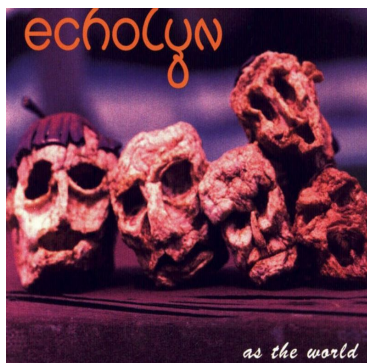
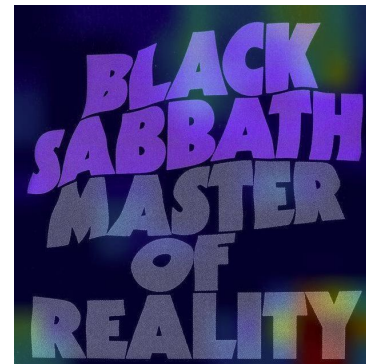
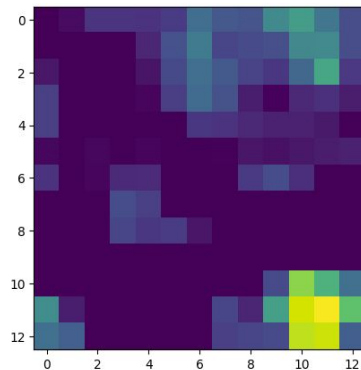
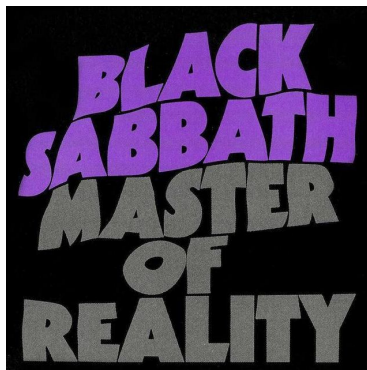
GradCAM (superimposed visualization) - Image 1



GradCAM (superimposed visualization) - Image 2



Extra Prog



Findings & Conclusions

- Metrics indicate we created an accurate CNN classification model that does not appear to be overfit
- Heat maps & GradCAM visualize activated aspects of the images to classify albums
- Prog rock albums identified by abstract patterns and symbols
 - Weird right corner?
- Other genre albums identify faces and words
- Questions of hidden overfitting from scaling?

Why Deep Learning?

- Deep Learning is necessary since image classification with CNNs is not something that can be done with simple models.
- Simple models cannot take in so much data with specific features
- Multiple layers and training required for this classification
- Depth of model is important in understanding the accuracy in detecting layered/deep architectures.
- Grad-CAM/heatmaps to detect most important features for specific details/features also use deep learning.

Questions?