Improving CNN model for weather image recognition

Team members
Yiheng Ye
Scott He
Debalina Chowdhury

Problem: Weather Image Classification

- Train specialized CNN models on complex real-world weather image recognition dataset and improve the prediction accuracy.

- Develop better CNN models that can be used under actual images that include other objects (cars, houses, trees, etc.) to classify the current weather in a given image.

True: hail
Predicted: hail

True: hail
Predicted: hail

True: snow
Predicted: snow

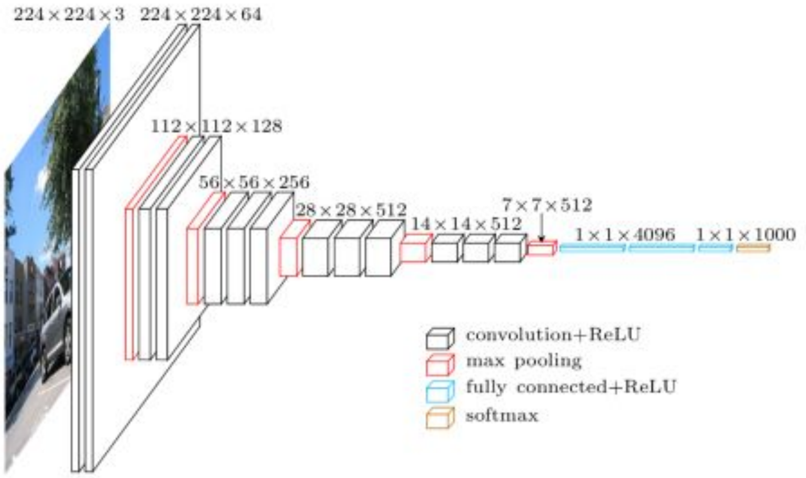
Data Set: Weather Image Recognition

- Weather image data collected with 11 different weathers including rain, snow, hail, glaze,...dew, etc.
- Total size is 636.73MB with 6862 files of images.
- Link:https://www.kaggle.com/jehanbhathena/weather-dataset

Solution: Existing Model Comparison and Creating Our Own Model

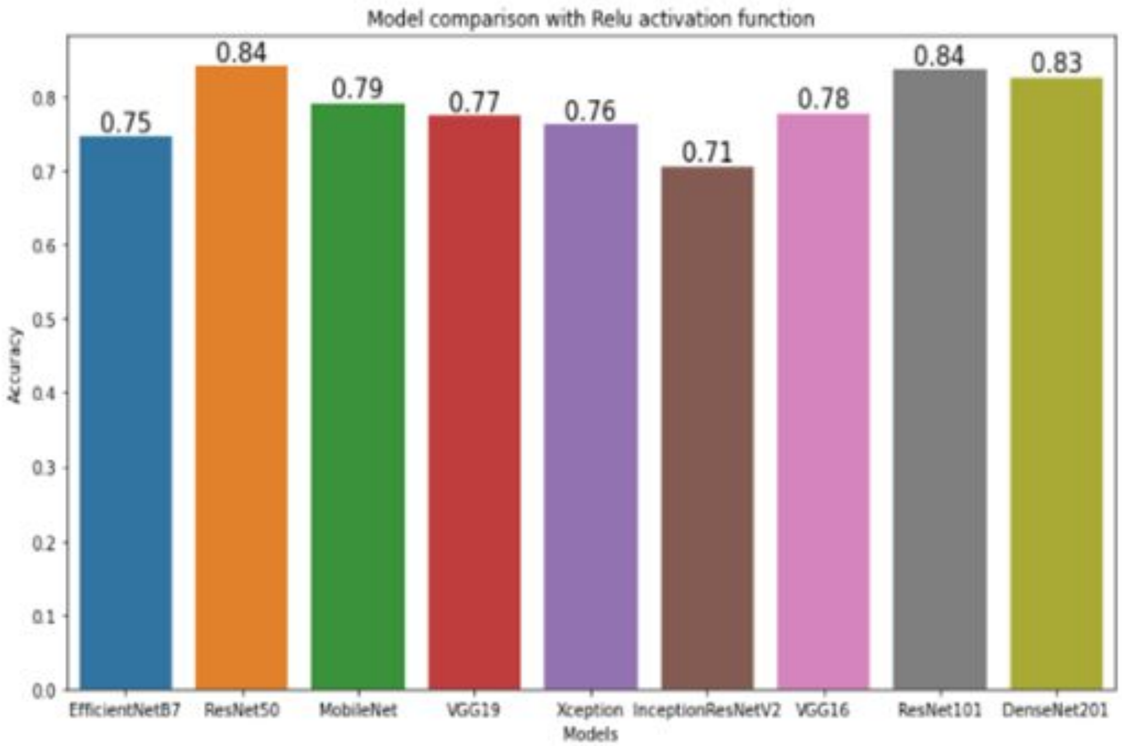
- Modifying VGG-16
- Making Comparison with different models and activation functions to define a proper baseline
- Observing and learning other improved model like MeteCNN
- Creating our own modified VGG-16 with similar changes like other improved model.

VGG16 Model



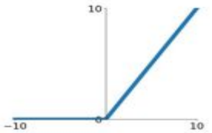
- VGG16 has 13 convolutional layers, and 3 fully connected layers, for a total of 16 layers with weights.
- Every convolutional layer uses a 3x3 filter with stride 1 and ReLU as its activation function.
- Every maxpool layer uses a 2x2 filter with stride 2.
- Has approximately 138 million parameters in total.

Comparison of models using ReLU activation function



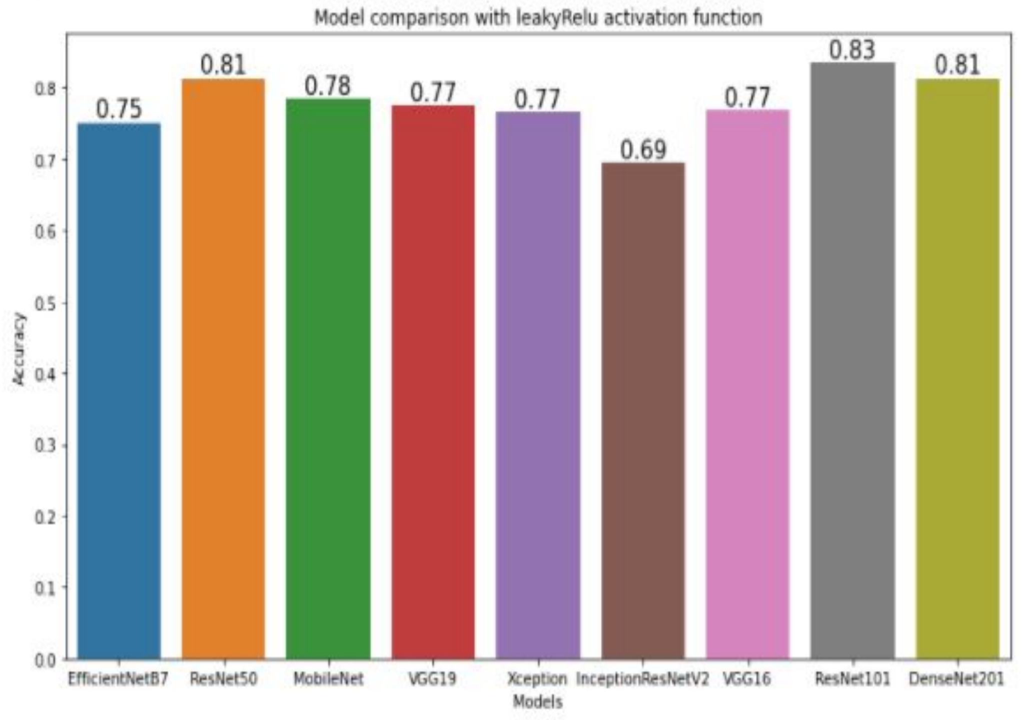
- Firstly, we tried our models with the most commonly used activation function ReLU.

f(x) = max(0,x)

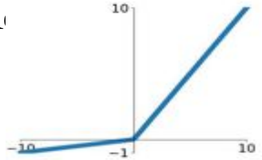


- As we compare the performance of different CNN models on test dataset, we can observe that **Resnet50** performs the best with a test accuracy of **84%** using **ReLU** activation functions for hidden layers and softmax function for output layer.

Comparison of models using leakyReLU activation function



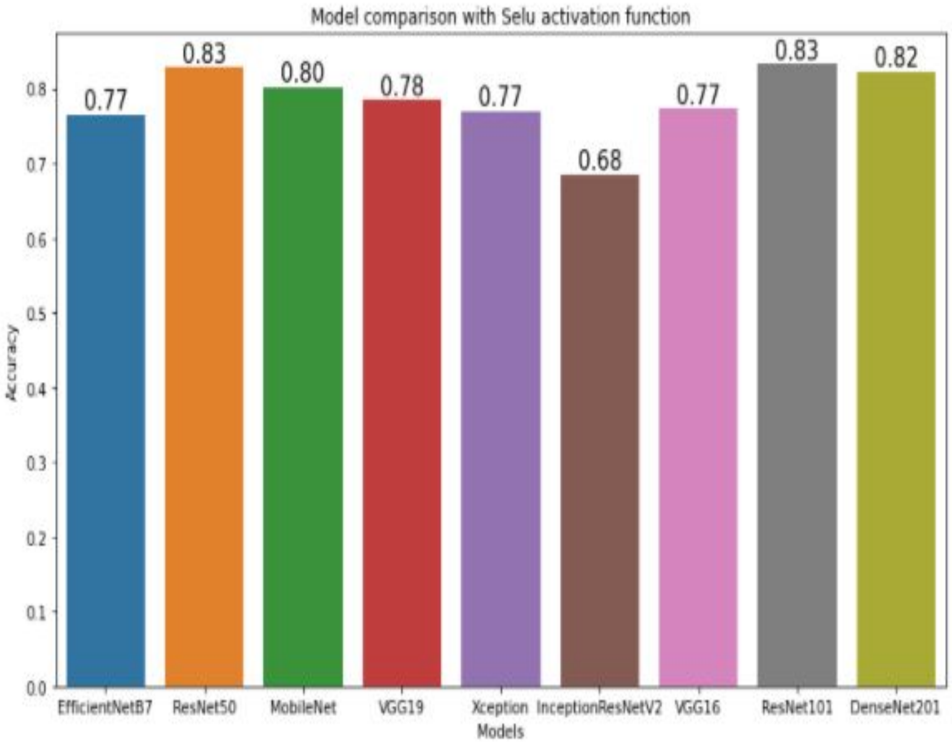
- Next we tried our model with a variant of ReLU function i.e. leakyR



f(x)=max(0.1x,x)

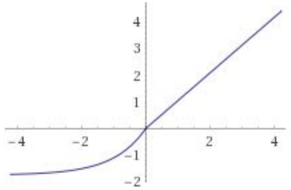
- As we compare the performance of different CNN models on test dataset, we can observe that **Resnet101** performs the best with a test accuracy of **83%** using **leakyReLU** activation functions for hidden layers and softmax function for output layer.

Comparison of models using SeLU activation function



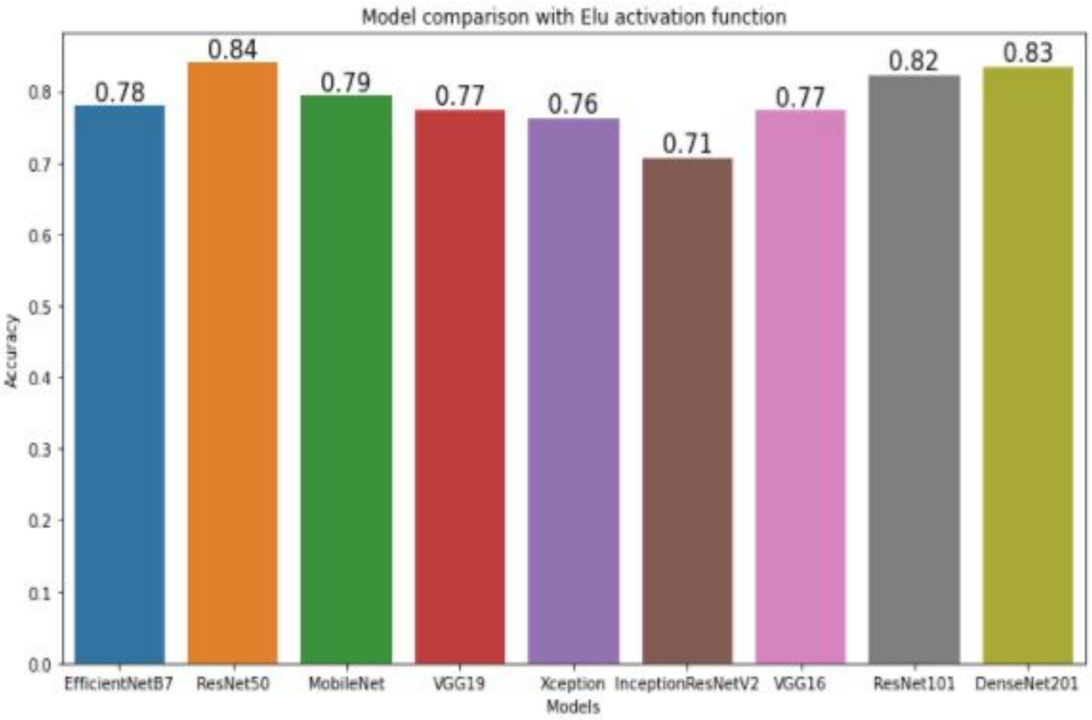
- Now, we tried another version of ReLU i.e. SeLU.

  $f(x) = \lambda x$ if $x>0$
  $f(x)=\lambda\alpha(e^x-1)$ if $x<=0$



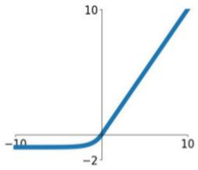
- As we compare the performance of different CNN models on test dataset, we can observe that **Resnet101** and **Resnet50** both perform the best with a test accuracy of **83%** using **SeLU** activation functions for hidden layers and softmax function for output layer.

Comparison of models using ELU activation function



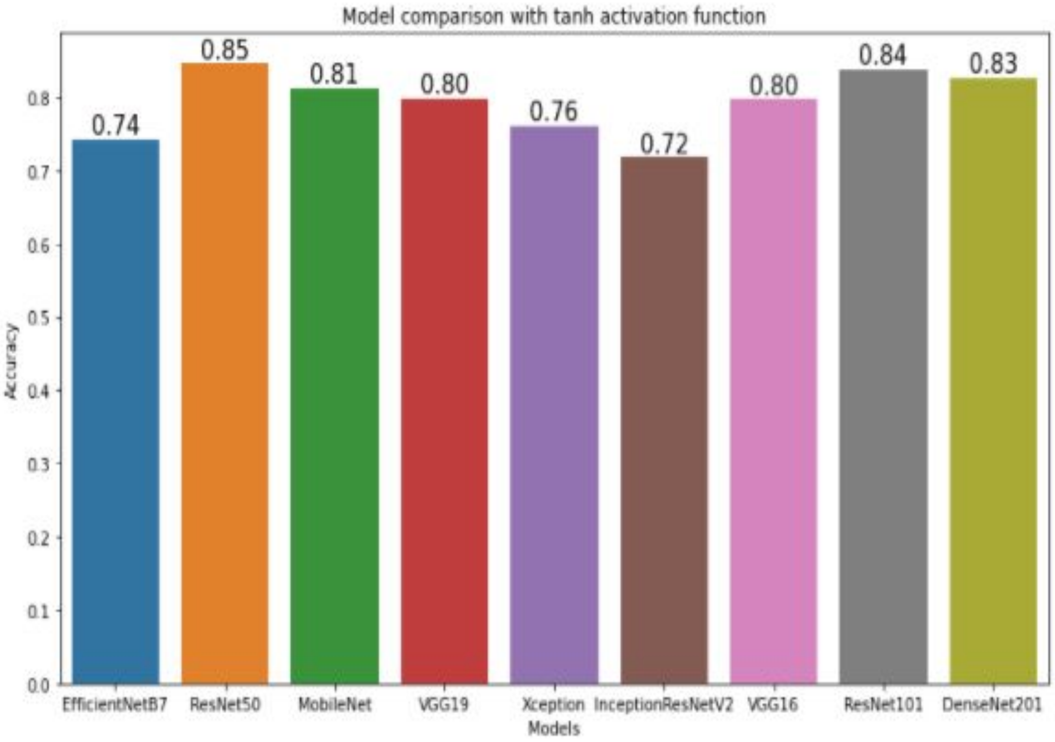
- Now, we consider ELU as the activation function.

  $f(x) = x \quad \text{if } x>=0$
  $f(x)=\alpha(e^x-1) \quad \text{if } x<0$



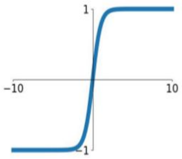
- As we compare the performance of different CNN models on test dataset, we can observe that **Resnet50** performs the best with a test accuracy of **84%** using **ELU** activation functions for hidden layers and softmax function for output layer.

Comparison of models using tanh activation function



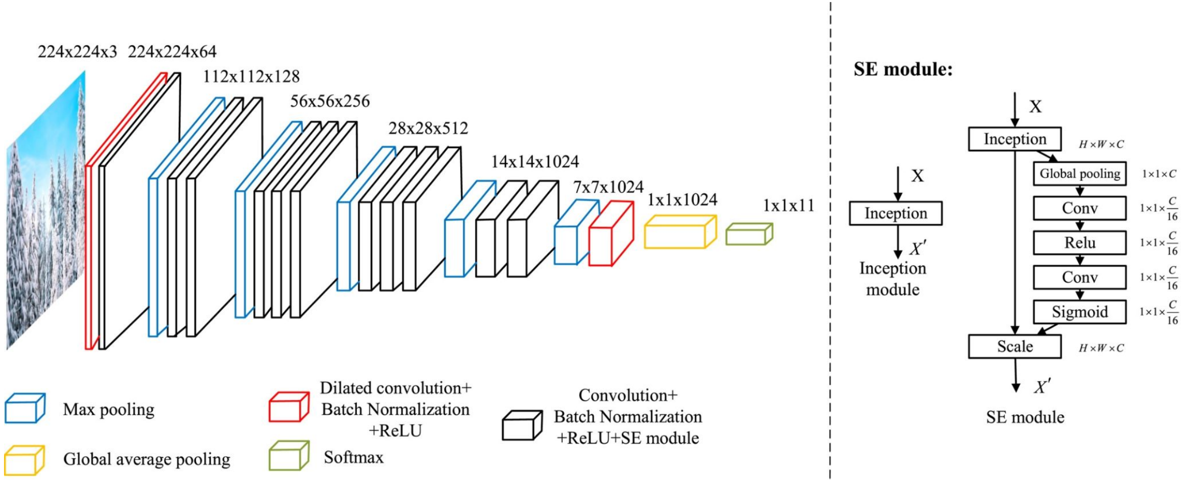
- Lastly, we tried the models with tanh activation function.

$$f(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}$$



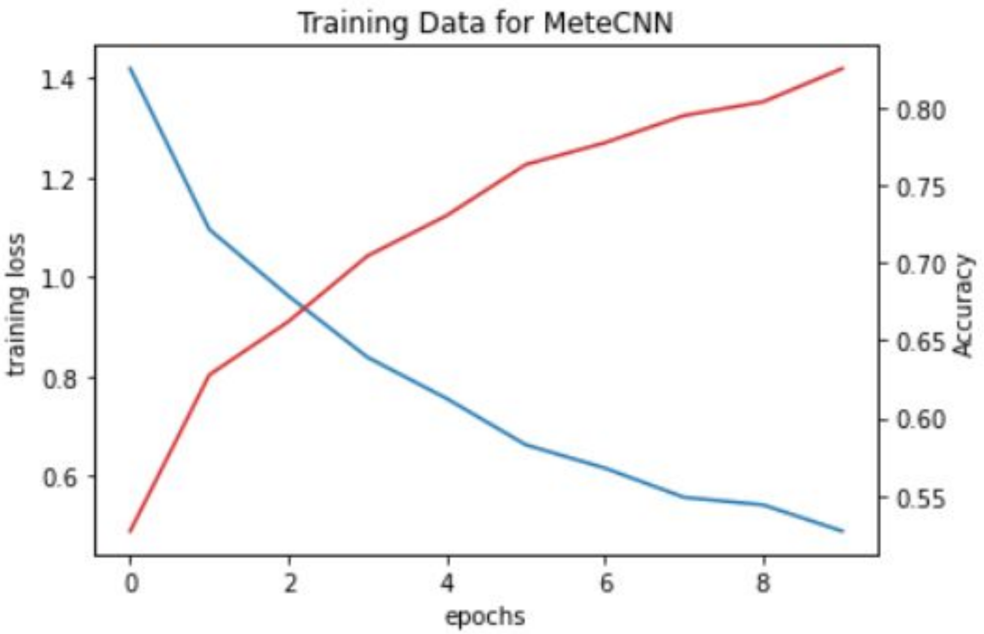
- As we compare the performance of different CNN models on test dataset, we can observe that **Resnet50** performs the best with a test accuracy of **85%** using tanh activation functions for hidden layers and softmax function for output layer.
- We can also observe that **85%** is the **highest test accuracy** achieved over all the models so far.

MeteCNN Model Architecture



- Based on VGG16 architecture. The two fully connected layers at the end are replaced with a global average pooling layer.
- In addition, there are two layers featuring dilated convolution, and batch normalization is done after every convolution.
- Furthermore, it uses a Squeeze-Excitation(SE) module in its convolutional layers to improve performance.

MeteCNN-First Look



	precision	recall	f1-score	support
0	0.83	0.79	0.81	100
1	0.83	0.55	0.66	154
2	0.88	0.52	0.65	337
3	0.67	0.78	0.72	202
4	0.44	0.41	0.42	166
5	0.43	0.83	0.57	145
6	0.66	0.66	0.66	206
7	0.82	0.80	0.81	186
8	0.42	0.65	0.51	138
9	0.49	0.33	0.40	171
10	0.56	0.79	0.66	57
accuracy			0.62	1862
macro avg	0.64	0.65	0.62	1862
weighted avg	0.67	0.62	0.62	1862

The validation accuracy is very low compared to other methods!

Our Solution: Removed SE module

	precision	recall	f1-score	support
0	0.88	0.90	0.89	100
1	0.86	0.70	0.77	154
2	0.89	0.66	0.75	337
3	0.75	0.83	0.79	202
4	0.48	0.44	0.46	166
5	0.60	0.82	0.70	145
6	0.70	0.79	0.74	206
7	0.81	0.86	0.83	186
8	0.44	0.74	0.55	138
9	0.60	0.34	0.43	171
10	0.60	0.61	0.61	57
accuracy			0.70	1862
macro avg	0.69	0.70	0.68	1862
weighted avg	0.72	0.70	0.69	1862

Validation accuracy increases a lot in exchange for a slight loss in training performance

Future Exploration: More Epochs Trained?

	precision	recall	f1-score	support
0	0.92	0.89	0.90	100
1	0.80	0.73	0.76	154
2	0.86	0.68	0.76	337
3	0.92	0.85	0.88	202
4	0.41	0.60	0.49	166
5	0.40	0.96	0.57	145
6	0.75	0.83	0.79	206
7	0.74	0.82	0.78	186
8	0.77	0.30	0.43	138
9	0.50	0.08	0.13	171
10	0.59	0.79	0.68	57
accuracy			0.68	1862
macro avg	0.70	0.68	0.65	1862
weighted avg	0.72	0.68	0.66	1862

Validation metrics for our own model (20 epochs)

	precision	recall	f1-score	support
0	0.89	0.86	0.87	100
1	0.83	0.52	0.64	154
2	0.75	0.81	0.78	337
3	0.87	0.77	0.82	202
4	0.45	0.45	0.45	166
5	0.72	0.70	0.71	145
6	0.77	0.83	0.80	206
7	0.82	0.81	0.81	186
8	0.46	0.77	0.58	138
9	0.63	0.43	0.51	171
10	0.63	0.74	0.68	57
accuracy			0.71	1862
macro avg	0.71	0.70	0.69	1862
weighted avg	0.72	0.71	0.70	1862

Validation metrics for MeteCNN (20 epochs)

Conclusion

- Current image classification models have already been doing well in classifying weather image.

- We can improve current models by adjusting activation function from ReLU to Tanh if possible.

- MeteCNN is not always working ideally well as in the case of our dataset.

- We can improve MeteCNN, or at least obtain a more efficient model, by removing SE module.

References

[1] Haixia Xiao,Feng Zhang,Zhongping Shen,Kun Wu,Jinglin Zhang (2021) , Classification of Weather Phenomenon From Images by Using Deep Convolutional Neural Network https://agupubs.onlinelibrary.wiley.com/doi/10.1029/2020EA001604

[2] Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. In *Computer Vision and Pattern Recognition*. arXiv preprint arXiv:1409.1556