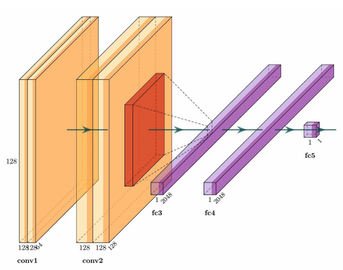
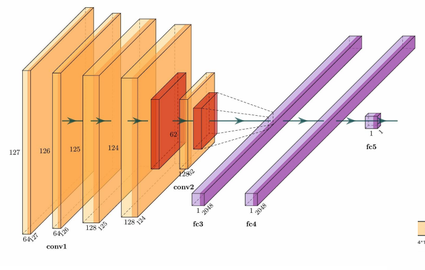
Hotdog/not hotdog

Yu Zhang(s230000), Lin Zhu(s232291), Dong Yun(s232293)
Cong Jin(s232254), Yaying Cai(s232284)

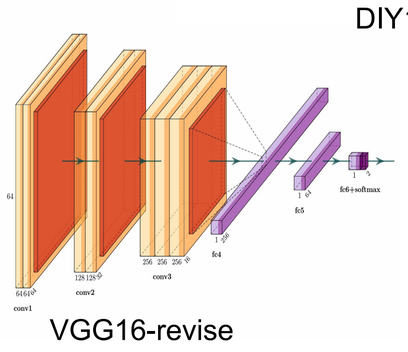
Architecture



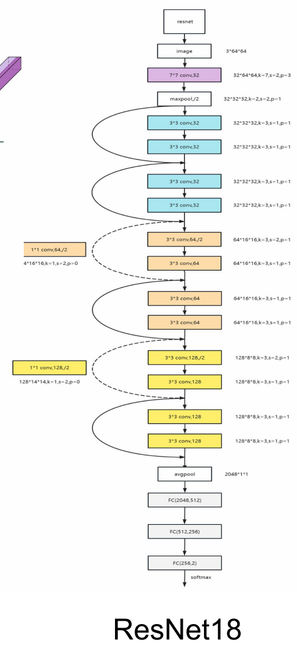
DIY2



DIY1



VGG16-revise



ResNet18

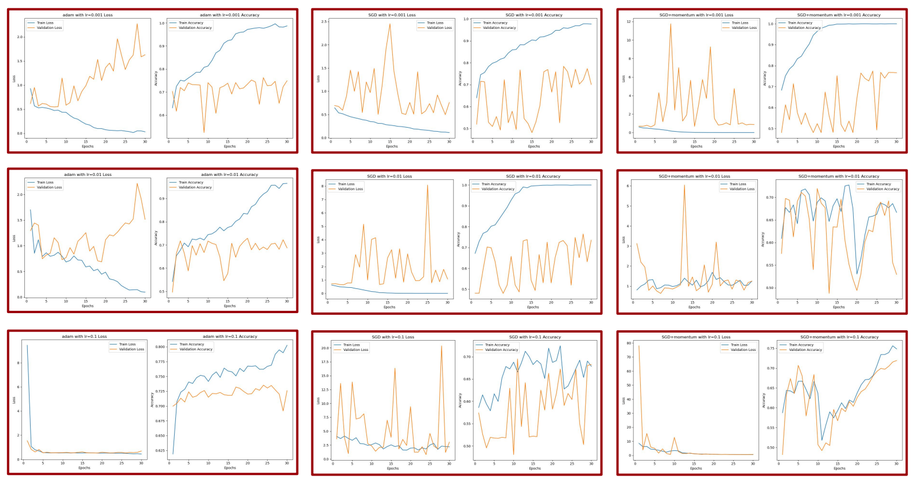
Initially, we design our own CNN architectures (**DIY1** and **DIY2** etc.). We try to explore different combinations. For example, 2 layers of 3×3 convolutional layers, 4 layers of 2×2 convolutional layers, 1 layer of 5×5 convolutional layers, the replacement of these three. But the performance is ordinary, so we try to use some famous architectures.

We experimented with **ResNet18** for our architecture because of its impressive performance in image classification tasks. However, during training, we noticed that the ResNet model was overfitting to the training data, even after implementing regularization techniques.
We also decided another famous architecture **VGG16**, but it's easy to vanish gradients. As a result, we cut off the back layers and make the **VGG16-revise** for our architecture. After adopting VGG16-revise, we noticed a significant reduction in overfitting and an improvement in validation accuracy.

Training

In this part, we train the ***VGG16-revise*** with different optimizers and learning rates. Below is the our recording. **SGD** seems to be the most efficient.

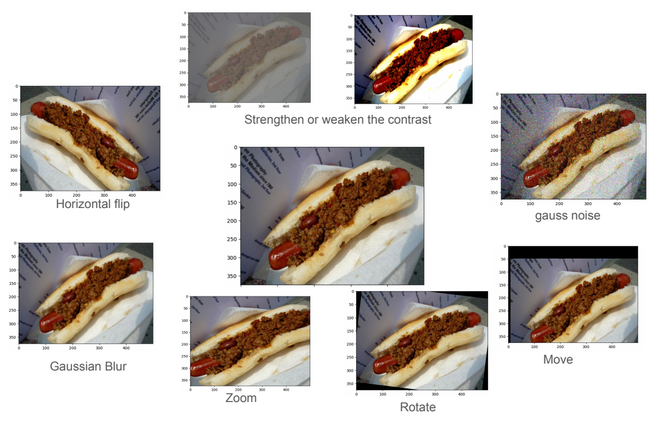
Adam SGD SGD+momentum



Model	Training Time (s)	Best Validation Accuracy
SGD with lr=0.001	120.2702551	0.764769066
SGD+momentum with lr=0.01	120.4992628	0.722878625
SGD with lr=0.01	120.7053332	0.76047261
SGD with lr=0.1	120.9691715	0.723952739
SGD+momentum with lr=0.1	121.1612096	0.729323308
SGD+momentum with lr=0.001	121.7241342	0.758861439
adam with lr=0.01	130.22053	0.715896885
adam with lr=0.1	132.5621243	0.729323308
adam with lr=0.001	134.166966	0.747046187

Besides, we added **batch normalization** to our model during the training. By implementing this, the model converged more quickly during training, and demonstrated improved generalization with a reduction in overfitting. Especially when the model has too deep layers, we need to add batch normalization which helps prevent gradients vanishing.

Data Augmentation



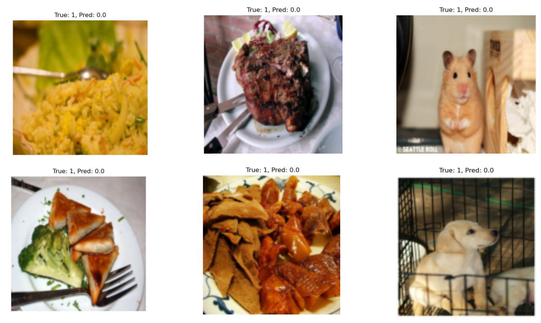
	Initial Data	Augmented Data
VGG16_revise	76.5%	81.4%
ResNet18	77.9%	79.8%

We used data enhancement techniques during training to help the model learn better. The exact technique is shown in the screenshot on the left.
From above results, we noticed that data augmentation improved the accuracy of both models.

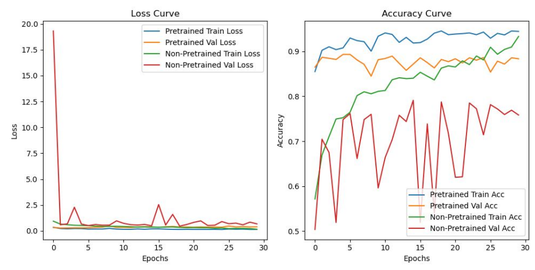
Accuracy & Misclassified Images

The accuracy of our network is shown below. The majority of misclassified Images is labeled **food,** then the one with the label **pets** is the second most.
We think that Food images with plates or red dish being misclassified via shortcut learning sounds reasonable. But we are not sure about why pets and people be misclassified.

Architecture	Data	Best Acc
DIY1	initial data	0.7127
DIY2	initial data	0.7234
VGG16	initial data	gradient vanishing
VGG16_revise	initial data	0.7647
ResNet18	initial data	0.7786
VGG16_revise	*augmented*	0.8142
ResNet18	*augmented*	0.7984



Transfer Learning



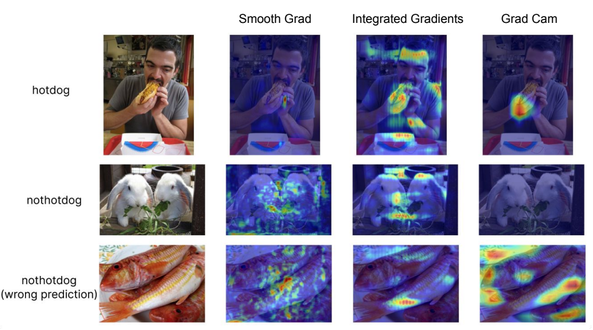
In this part, we apply ResNet18 (which is pretrained in the dataset ImageNet) into our dataset (no augmentation).
Comparing the pretrained ResNet-18 and non-pretrained ResNet-18, we find that the accuracy increases a lot.

However, we think it works because the hot/not hotdog task is too easy. If we want to train a more complex model, the distribution shift will affect the performance. As a result, we may pay more attention to **fine tuning** when using transfer learning on complex tasks.

Saliency Maps & Shortcut Learning

In this set of Saliency maps, the three algorithms highlight the areas the model focused on.

Smooth Grad produces weaker and more diffuse activations because it relies on input gradients and noise averaging, which may not effectively represent higher-level features.

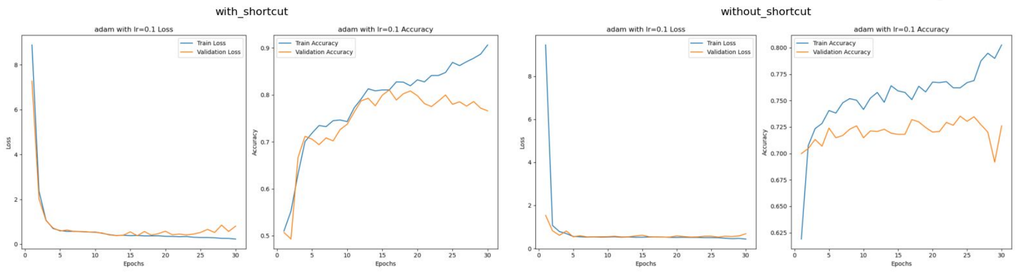


In contrast, Integrated Gradients and Grad CAM aggregate gradient information more effectively across the network's layers or along meaningful paths, resulting in clearer and more focused saliency maps.

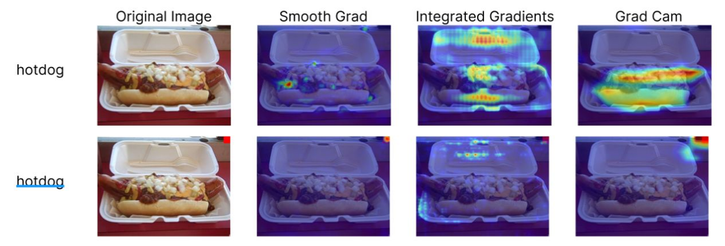
Shortcut Learning（additional part)

We've added a red pixel to the top right corner of the image for 60% of all hotdog data (without augmentation), trying to figure out whether the model will have shortcut learning.

According to below results, we find that model trained in the revised dataset has higher accuracy.



To further confirm if the model has shortcut learning, we plot saliency maps. Highlight in the upper-right corner shows that the model did use the shortcut learning.



Due to the limited time, we didn't go any further into how to eliminate shortcuts and it may be interesting and challenging.

Help of generative model

1.Help us search related literatures and tell us famous architectures since 2014, through which we choose 2 CNN-based architectures VGG and ResNet.
2.Help us debug the program
3.Help us make the plots of architectures.

References

[1] Koonce, Brett, and Brett Koonce. "Vgg network." Convolutional Neural Networks with Swift for Tensorflow: Image Recognition and Dataset Categorization (2021): 35-50
[2] K. He, X. Zhang, S. Ren, J. Sun. “Deep Residual Learning for Image Recognition”. 2015. arXiv:1512.03385
[3]Smilkov, Daniel, et al. "Smoothgrad: removing noise by adding noise." *arXiv preprint arXiv:1706.03825* (2017).
[4]Selvaraju, Ramprasaath R., et al. "Grad-cam: Visual explanations from deep networks via gradient-based localization." *Proceedings of the IEEE international conference on computer vision*. 2017.