Identifying AI-generated Images

G008(s1925715, s1904845, s1923846)

Abstract

With the popularity of diffusion models such as stable diffusion, images generated by AI are gradually filling the internet. And with the development of diffusion models, the photorealism of generated images becomes astonishing, sometimes they can even deceive human eyes. Therefore, how to detect the images generated by diffusion models has gradually become a hot topic. We found that there are a lot of studies on detecting images generated by GANs models, and good accuracy has been obtained through data augmentation, image spectrum extraction and other methods. However, there are relatively few studies on detecting images generated by diffusion models, and some studies have found that the detector has poor generalization ability when detecting images generated by different types of models. So in our study, we focus on detecting the realistic style images generated by the stable diffusion model. We will compare the performance of different models and the effect of different data preprocessing methods. As well as using grad-cam and convolution layer vis to improve the interpretability of the model and to analyse the defects of the DMs (diffusion models)-generated images.

1. Introduction

Image synthesis has been a controversial area in the realm of Artificial Intelligence (AI), it relieves human artists' burden by automatically creating high-quality images, but it can be misused for a wide range of malicious purposes. The notorious deep fakes have harmed countless victims by synthesizing deceptive disinformation or pornography without consent. Recent achievements in AI have allowed algorithms to generate more realistic contents(Suwajanakorn et al., 2017) (Elgammal et al., 2017) (Kim et al., 2018), identifying AI manipulated contents manually becomes a costly task, therefore, people are investigating the automation of this task.

To mitigate ethical issues, many sophisticated studies focused on detecting fake images or videos of human faces (Agarwal et al., 2019) (Matern et al., 2019) (Liu et al., 2020) (Li et al., 2018). These methods are task-specific and some involve the exploitation of human physiology, they may not perform well on other datasets. Recent techniques allow tools like Stable-diffusion and Dalle to synthesise pictures that are not necessarily human faces, the malicious usage of these tools are not frequently reported, but ethical issues still exist.

In 2018, an AI artwork, a portrait of Edmond Belamy, was sold for over $69 million, the profit was not shared with data providers [1]. In fact, human arts as AI's training data are rarely used after receiving a formal acknowledgement, developers' right of gaining profit remains questionable (Cetinic & She, 2022). Academia is still in discussion about mitigating the copyright issue. Nowadays, tagging AI-generated pictures explicitly becomes a regulation on many web platforms. There are researches about detecting synthetic images, (Corvi et al., 2022) (Gragnaniello et al., 2021) (Sha et al., 2022) existing online APIs like Hive and illuminate can identify synthetic images reliably. However, none of them generalizes well, the model trained on generative adversarial networks (GAN) generated images perform worse on diffusion models (DM) generated images, and vice versa. Illuminarty is a special case where it is only good at identifying synthetic anime-style images.

In addition, we also found that in most studies, the researcher's choice of model is very limited. Most of them choose to use the ResNet model pre-trained on ImageNet as the baseline model (Wang et al., 2019) (Sha et al., 2022) (Zhang et al., 2019), and only a few people tested the transformer model (Somepalli et al., 2022). Therefore in our research, we aim to compare the performance and generalization ability of various models on this classification task.

In addition to this, to the best of our knowledge, there is no research on the interpretability of detectors in the current study. Also, analysis for AI-generated images is only available from the perspective of the DFT spectrum and frequency domain. In our study, we have, for the first time, analysed the defects of AI-generated images in terms of the interpretability of the model.

2. Data set and task

Our research hypothesis is that we can use transfer learning and pre-trained models to identify AI-generated images (specifically, generated by the stable diffusion model). We expect the trained model to have an accuracy higher than 90%, and we could observe some trends by analyzing the model's gradient.

[1]https://www.christies.com/features/A-collaboration-between-two-artists-one-human-one-a-machine-9332-1.aspx

ImageNet is a well-known dataset of images, the classes of images are arranged in a hierarchical manner according to a language dataset, WordNet. Each class in ImageNet can be described with a word sense in WordNet. (The complete imageNet consists of 20 thousand classes with a total of more than 14 million pictures)There are over 20 thousand classes in total, and training models on the entire ImageNet can be extremely computationally expensive. Therefore, a subset of ImageNet called ImageNet Large Scale Visual Recognition Challenge (ILSVRC) 2012 is often used instead. Due to the limitation of GPU memory, we only used a small part of ImageNet-1k, which is a subset of ILSVRC 2012.

Stable ImageNet-1K is a dataset generated by stable diffusion, it consists of 1000 classes of images, and each class has 100 images. Each class corresponds to a class in ImageNet-1K. Since stable diffusion is a text-to-image generator, it may create faults when the class name is ambiguous, for instance, "Kite" can be a flying object made by light frame and paper, and it can also be the name of a female, thus images are generated with both word senses. But our detector is just a binary detector that detects whether the image is generated by AI, so the semantic content of the image does not confuse the model.

Two datasets are very unbalanced in size, the Stable ImageNet-1k has a size of less than 10GB, whereas ILSVRC 2012 has a size of over 150GB. Assuming both datasets have a similar average image resolution, a model trained with both entire datasets that always classify images as real will have a very high accuracy. Thus we have randomly selected 100 classes and each consists of 100 images from both datasets to run our first feasibility test. The resulting trained vanilla model performs decently well, we then carried out further experiments with another dataset.

There are 3 reasons for using a different dataset: (1) we have resized each image into 256 * 256 pixels before feeding it into the neural network, where this process is computationally demanding, thus we resized the image at the preprocessing stage. (2) all generated images have a resolution of 512 * 512, whereas ImageNet-1k is chaotic, the image resolution can vary from 2816*2112 pixels to 90*90 pixels, and the ratio between height and width is not fixed either. The resized image may have some level of distortion/blur and a neural network could potentially learn this unrelated feature. (3) The training curve of the experimental model is unstable, and the performance could be improved potentially, so we decided to use a larger dataset.

The dataset later used in the project consists of all images from Stable ImageNet-1k, and we want to choose 100 suitable images from each class from ImageNet-1k. Firstly, the pre-trained model is capable of identifying the class of an image, if each class does not have a balanced number of images from both datasets, the trained model can be biased. For instance, there are 190 real images of a frog, 100 generated images of a frog, 10 real images of a bird, and 100 generated images of a bird. The model is prone to predict the bird image as AI-generated, regardless of visual features, even if the number of real and fake images are the same. We also attempt to minimise the biases by selecting images with a size greater than 256 * 256 and with a width-height ratio close to 1. Although some classes in ImageNet-1k have really low image quality, and we cannot find 100 suitable images from that class. The preprocessed dataset has 100000 generated images and 99800 real images.

3. Methodology

3.1. Data augmentation

We have applied 3 types of data augmentations: random crop, random flip (vertical or horizontal), and random rotation (10 degrees). These are common techniques, previous works have shown that data augmentations are beneficial to the model performance by reducing overfitting (Shorten & Khoshgoftaar, 2019) (Krizhevsky et al., 2017).

3.2. Baseline Models

Since there isn't much existing research on the topic of classifying AI-generated Images, we further proposed two types of basic neural network models for experimenting with the effects of convolution and non-convolution models, as well as comparing the performances with our proposed model. We design a 3-layers vanilla model (aka feed-forward model) and a 5-layer convolution model as our baseline model.

The vanilla model is just a 3-layers feed-forward neural network which was originally proposed by (Svozil et al., 1997). The model contains 3 fully connected layer and the first two fully connected layer is followed by ReLu activation and the last connected layer is followed by a log softmax activation. The details of our vanilla model structure are:

Vanilla:

- Linear(in_features=150528,out_features=128,bias=True)
- ReLu()
- Linear(in_features=128,out_features=32,bias=True)
- ReLu()
- Linear(in_features=32,out_features=2,bias=True)
- Log_Softmax()

The convolution model structure is similar to LeNet(Lecun et al., 1998) which is a convolution model that has two convolution layers and 3 fully connected layers. Each convolution layer is followed by a Max Pooling operation and each fully connected layer is followed by a ReLu activation function. The last connected layer is followed by a log softmax activation. The details of our convolution model structure are:

Convolution:

- Conv2d(in_channel=3,out_channel=6,kernel_size=5)
- MaxPool2d(,kernel_size=2, stride=2)
- Conv2d(in_channel=6,out_channel=16,kernel_size=5)
- MaxPool2d(,kernel_size=8, stride=8)
- Linear(in_features=2704,out_features=128,bias=True)
- ReLu()
- Linear(in_features=128,out_features=32,bias=True)
- ReLu()
- Linear(in_features=32,out_features=2,bias=True)
- Log_Softmax()

For both baseline models, we use CrossEntropyLoss as the calculation of the loss and SGD as the update of gradient descent, where lr = 0.001 and momentum = 0.9.

3.3. Transfer Learning

We use transfer learning because of its ability to use pre-trained models to solve new tasks with a limited amount of training data. When it comes to image classification tasks, ImageNet is one of the most widely used benchmark datasets, containing millions of labelled images across more than 1,000 object classes. By using transfer learning on ImageNet-based tasks, we can leverage the knowledge and representations learned by networks on this large dataset. Rather than training a model from scratch, starting with a network that is similar to a trained network will achieve higher accuracy and faster convergence with less training data. Therefore, choosing to use transfer learning on a model that is pre-trained on ImageNet can be an efficient strategy to solve new image classification tasks with limited data.

In our experimental design, we chose ResNeXt and Swin Transformer models as pre-training models. These two models are not the highest accuracy models currently available, as the top three models on ImageNet are BASIC-L, CoCa and Model Soups (BASCI-L), all of which are transformer models and have a number of parameters from 2100M to 2400M. On the one hand, this is an extremely large model to train, and on the other hand, given the size of the models and the complexity of the task we are studying, we believe that the capacity of the ResNeXt and SwinT models is sufficient to solve this binary image classification task without requiring too much training time. Also, we chose ResNeXt as a representative of the CNN models and SwinT as a representative of the transformer models, in order to compare the performance of the two types of models on this task. The following is the implementation of transfer learning based on these two models.

3.3.1. ResNeXt_v1

The specific version of the pre-trained model we have chosen is ResNeXt101_64X4D, and the final output layer of the model is structured as a mean pooling layer followed by a fully connected layer, as follows.

(avgpool):

AdaptiveAvgPool2d(output_size=(1, 1))

(fc):

Linear(in_features=2048,out_features=1000,bias=True)

We first freeze the **avgpool** and all previous layers with **param.requires_grad = False** so that the pre-trained weights are not changed in later training. We then change the original fully connected layer to a two-dimensional linear classifier, in order to fit our binary classification task.

Later in the training process, we use CrossEntropyLoss as the calculation of the loss and SGD as the update of gradient descent, where lr = 0.001 and momentum = 0.9.

3.3.2. ResNeXt_v2

Considering that training only one fully connected layer may not be sufficient to fit the task, we also built a second model based on ResNeXt with more trainable parameters to improve the performance of the model using.

First, all layers before **layer4** of the original model were frozen using **param.requires_grad = False**. The reason for this is that the last few layers of the CNN model are directly related to the specific outcome of the task. The original pre-trained model was classifying images into specific categories, but our task only required classifying images as being or not being AI-generated and did not need to know the specific category information of the things in the images. Therefore the weights of the layers further back in the model are less related to our task. Because we chose to adapt the model to the new dataset and the new classification task, I chose to unfreeze all the weights of the entire **layer4** located at the end of the model.

layer4 is a sequence layer consisting of 3 bottleneck layers, with the structure shown in Figure 1.

On the left is the overall structure of layer4, and on the right is the structure of one of the bottlenecks after expansion. After layer4 is the final output layer consisting of three fully connected layers.

Loss is calculated in the same way as before and the gradient descent update algorithm is CrossEntropyLoss and SGD, respectively, where SGD has lr = 0.001 and momentum = 0.9.

3.3.3. Swin Transformer

The implementation of SwinT-based transfer learning is similar to that of ResNeXt. SwinT model has a different architecture from the CNN model, but the output layer at the end has a similar structure, as follows:

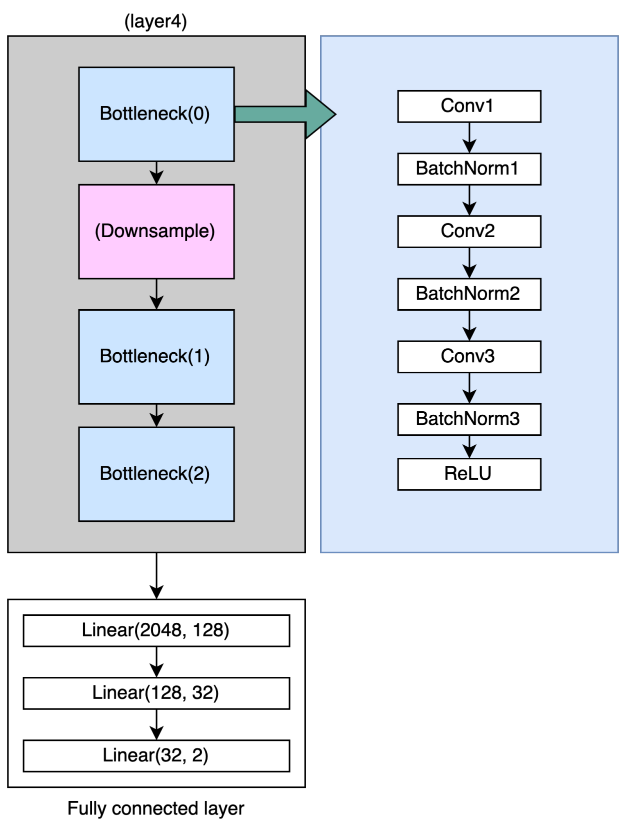

Figure 1. The structure of final layers in ResNeXt.

(norm): LayerNorm((1024,),eps=1e-05)

(permute): Permute()

(avgpool): AdaptiveAvgPool2d(output_size=1)

(flatten): Flatten(start_dim=1,end_dim=-1)

(head): Linear(in_dim=1024,out_dim=1000,bias=True)

The implementation is the same as before, we use **param.requires_grad = False** to freeze the weights of all layers before the last linear, and then the output dimension of the final linear layer is changed to 2.

For the loss function and gradient descent algorithm, the same choices were made as before, for CrossEntropyLoss and SGD, where SGD has an lr = 0.001 and a momentum = 0.9.

3.4. Guided Backpropagation Saliency Map

Guided Backprogration Saliency Map is a visualisation method used to interpret the decision-making process of a convolution neural network. It was first proposed by (Simonyan et al., 2013) and combined the idea of backpropagation and modify RELU in order to generate saliency maps that highlight the importance of different regions on inputs images given a class prediction.

The way of this technique works by first inputting an image into the network and calculating the gradient with backpropagation respect with to the given class label. A modified ReLu function is used during the process to ensure only positive gradients are propagated through the network and resulting in more clean and interpretable results. Once the backpropagation reaches the target convolution layer. It saves the gradient and normalizes it into a 0-1 region and generates a grayscale image. The generated image will be the saliency map of the target convolution layer, indicating which regions of image are being activated at the target convolution layer.

Besides using guided backpropagation to generate saliency maps for layer activation. We also use the gradient obtained from the backpropagation process to generate a positive saliency and negative saliency map. Where the positive saliency map is showing the regions that are positively correlated with the output and the negative saliency map is showing the regions that are negatively correlated with the output.

The Saliency Map could provide valuable insights into the decision-making process of the networks at different hierarchical levels and address the potential feature that helps to classifying the AI-generated images.

The implementation of the Saliency Map in our experiments is adopted from pytorch-cnn-visulization library(Ozbulak, 2022). We select the best-performing ResNeXt model on the test set for all the visualization and analysis. Our Saliency Map results will be shown in the Visualization and Result Analysis section at the end of the Experiment chapter.

3.5. Grad-CAM

Grad-CAM (Gradient Weighted Class Activation Mapping) is a technique widely used in computer vision to visualise the importance of different regions of an image to a particular classification decision made by a convolutional neural network (CNN) (Selvaraju et al., 2016). Grad-CAM uses the gradient of the target class output relative to the final convolutional layer of the CNN to produce a localisation map, highlighting the most important regions of the input image for the classification decision regions.

In our study, we implemented Grad-CAM to better understand the contribution of different regions of the input image to the final classification decision of our CNN-based model. Specifically, we use the last convolutional layer of the CNN model as the target layer of Grad-CAM. Given an input image, we perform forward propagation through the ResNeXt model to obtain the category scores of the predicted labels. We then back-propagate the class scores through the last convolutional layer and calculate the gradients of the class scores relative to the feature map. These gradients were pooled globally on average to obtain importance weights for each feature map. Finally, we multiply the feature maps by their corresponding importance weights and sum them to obtain the output of Grad-CAM.

To obtain a visual representation of the Grad-CAM output, we overlaid the Grad-CAM output onto the input image using a colour map. This allows us to identify the areas of the input image that contribute most to the classification decision. The resulting visualisation gives us an idea of the regions of the input image that the model focuses on when making classification decisions.

The implementation of the Grad-CAM in our experiments relies on the PyTorch grad-cam(Gildenblat & contributors, 2021) library. we have selected the best-performing ResNeXt model on the test set as the basis for the Grad-CAM. The target_layer is the last convolutional layer of the ResNeXt model, which is the last bottleneck layer in **layer4**. The target class is the highest-scoring result predicted by the model. The input images are processed differently from the pre-processing during training, we remove all the data augmentation methods and only keep the normalization part. Our Grad-CAM result will be shown in the **Visualization and Result Analysis** section at the end of the Experiment chapter.

4. Experiments

4.1. Experimental Objectives

Because detecting AI-generated images is a relatively new field, most of the detector models used in related research are limited to the ResNet model, and there has not been much research comparing the performance of different models on such a task. Also, as mentioned in the introduction chapter, the models are weak at generalising on this task, and the performance of the models degrades significantly when faced with images generated using different models like GAN models and diffusion models, and with different styles. Therefore we want to investigate through experiments how different models perform on this task, and whether using a transfer learning approach can allow models pre-trained on ImageNet to quickly adapt to new classification tasks. Finally, we will use Grad-CAM as well as convolution layer vis to enhance the interpretability of our models and to uncover the features of the DMs-generated images. The specific experimental objectives are as follows:

- Compare the performance of traditional CNN, MLP, ResNet18, ResNeXt101 and SwinT models on the task of detecting images generated by stable diffusion using the text-to-image method.

- Compare the impact of different training layers of transfer learning on model performance.

- Use the Grad-cam to improve the interpretability of the model for this task and analyse the model's detection principles.

- Use conv layer vis to improve the interpretability of the model and analyse the model's detection principles.

4.2. Experimental Setting

For baseline experiments, we designed two basic multi-layer neural networks as baseline models for comparison, including a vanilla neural network, and a regular convolution neural network. We also use some other well-known state-of-the-art methods including ResNetXt-101 and Swin Transformer as the pre-trained model for transfer learning. We modify the output of the last fully connected layer of the model to be a binary classifier. During the training process, we will freeze the weights of all other layers except the last fully connected layer as mentioned in the **Methodology** chapter.

To make the comparison with the pre-trained model, we have also trained a ResNet-18 model from scratch. Training a transformer or a deeper ResNet model can be extremely computationally demanding. As one of our main design decisions, transfer learning is computationally cheaper than training the whole model from scratch, thus we are not going to train a deep ResNet or transformer model from scratch for comparison.

For each model, we will use 60% of the dataset as the training set, and then adjust the hyperparameters on the validation set. Finally, we will pick the best-performing model in each architecture to compare their performance on the test set. We will compare the accuracy of models on the test set and their ability to generalize to unseen classes.

4.3. Visualization and Results Analysis

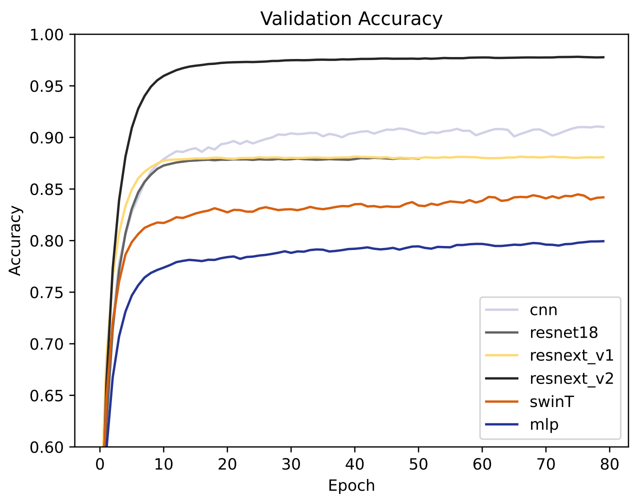

Figure 2. The Validation Accuracy of Models.

To make the visualization clearer, we have smoothed each curve and added 0 or 1 before each array of accuracy or loss.

As can be seen in Figure2, almost all models reached near convergence after 20 epochs of training, but the accuracy stayed at different levels. The best performer was the ResNeXt_v2 model using transfer learning, which achieved an accuracy of around 97%. The next best performer was the basic CNN model, which achieved around 90% accu-

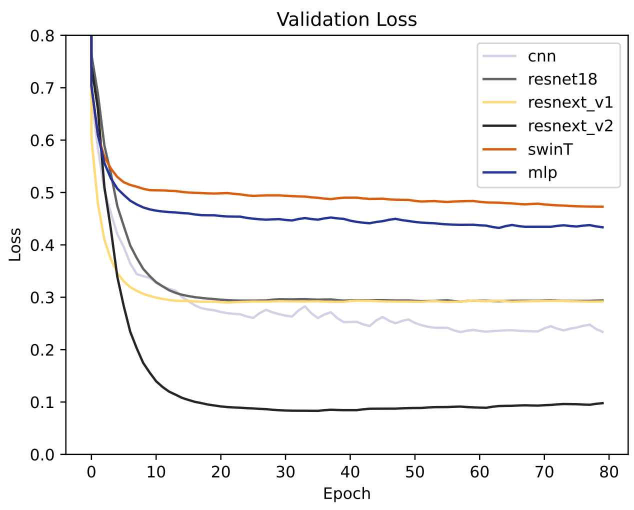

Figure 3. The Validation Loss of Models.

Model	Accuracy	Precision	Recall	F1 Score
Vanilla	0.7995	0.8024	0.7923	0.7973
Convolution	0.9131	0.9213	0.9036	0.9124
ResNet-18	0.8269	0.7775	0.9146	0.8405
ResNetXt_v1	0.8800	0.8761	0.8797	0.8779
ResNetXt_v2	**0.9729**	**0.9736**	**0.9721**	**0.9729**
Swin-T	0.8870	0.8846	0.8890	0.8868

Table 1. Evaluation results of models on the test set.

racy. This was followed by the ResNeXt_v1 model using transfer learning. This is a rather surprising result, as basic CNN is a much simpler model than ResNeXt_v1. We believe this is because unfreezing and training only the last layer of ResNeXt_v1 is not sufficient to adapt the pre-trained model to this new classification task, although the performance of basic CNN demonstrates that the classification task requires only a small model capacity. The same problem occurs with the swinT model, which also uses only one trainable layer.

The validation loss plot shows that the validation loss for all models decreases with the number of training epochs and then rapidly slows down to a constant level of around 20 epochs. This indicates that the models do not suffer from overfitting problems.

Regarding visualization outcomes, for real images 4, the activation within the neural network layer appears to be sporadically distributed throughout the entire image. This suggests that the majority of regions, encompassing both the object and its background, participate in the decision-making process. The positive saliency map and negative maps show a comparable trend, with the gradients predominantly dispersed across the entire image. The Grad-CAM, however, offers marginally distinct results, demonstrating that the network concentrates on the object when classifying an image as genuine.

In contrast, the visualization of fake images in 5 presents an entirely divergent pattern. Based on the Grad-CAM results, the network's focus is primarily on the background instead of the object. Additionally, the activation map and saliency map reveal that the image's background regions are involved in the decision-making process, with the majority of gradients concentrated in these areas. This finding contradicts conventional wisdom, as humans typically assess whether the object in the image is plausible; for instance in the 5, a bird should have two legs, the shadow should be accurate, and a fish should not possess three tails. The network seems to prioritize the background over the object, possibly due to the smooth backgrounds often found in fake images generated by the diffusion model, which facilitate detection.

In summation, the visualizations of authentic and counterfeit images allow for several conclusions. First, the background region is crucial for the classification of AI-generated images, as smooth backgrounds are commonly produced in these images, while genuine images exhibit more intricate and vibrant background patterns and colours. Second, it is plausible that AI-generated images possess a specific pattern when generating background colours; however, despite efforts to adjust the brightness and contrast of these images, no discernible pattern was identified. Third, the generated object within the image does not seem to play a significant role in the decision-making process. This observation could have negative implications for future research, as a considerable amount of AI-generated content contains objects that defy human common sense.

5. Related Work and Conclusions

5.1. Related Work

5.1.1. DE-FAKE

The task of their research is to detect and attribute fake images generated by text-to-image generation models (Sha et al., 2022). They propose two approaches, one is a ResNet18 model using only image as input. The other is a hybrid detector consisting of CLIP's image encoder and text encoder as feature extractors and a 2-layer multilayer perceptron classifier. The ResNet18 image-only model obtained an accuracy of 0.871 on the LD(Latent Diffusion)+Flickr30k (querying the prompts of Flickr30k to LD) dataset and 0.913 on SD(Stable Diffusion)+MSCOCO. Both image-only and hybrid detectors can achieve accuracy above 90% on datasets generated by a single model, but a hybrid detector has better generalisation ability between different image generation models.

5.1.2. DIRE for Diffusion-Generated Image Detection

In this study, they first evaluated the performance of the detector proposed in (Gragnaniello et al., 2021) which is based on ResNet50 on a GAN-generated image dataset, achieving almost perfect detection performance. They then evaluated the same detector again on DMs(Diffusion Models)-

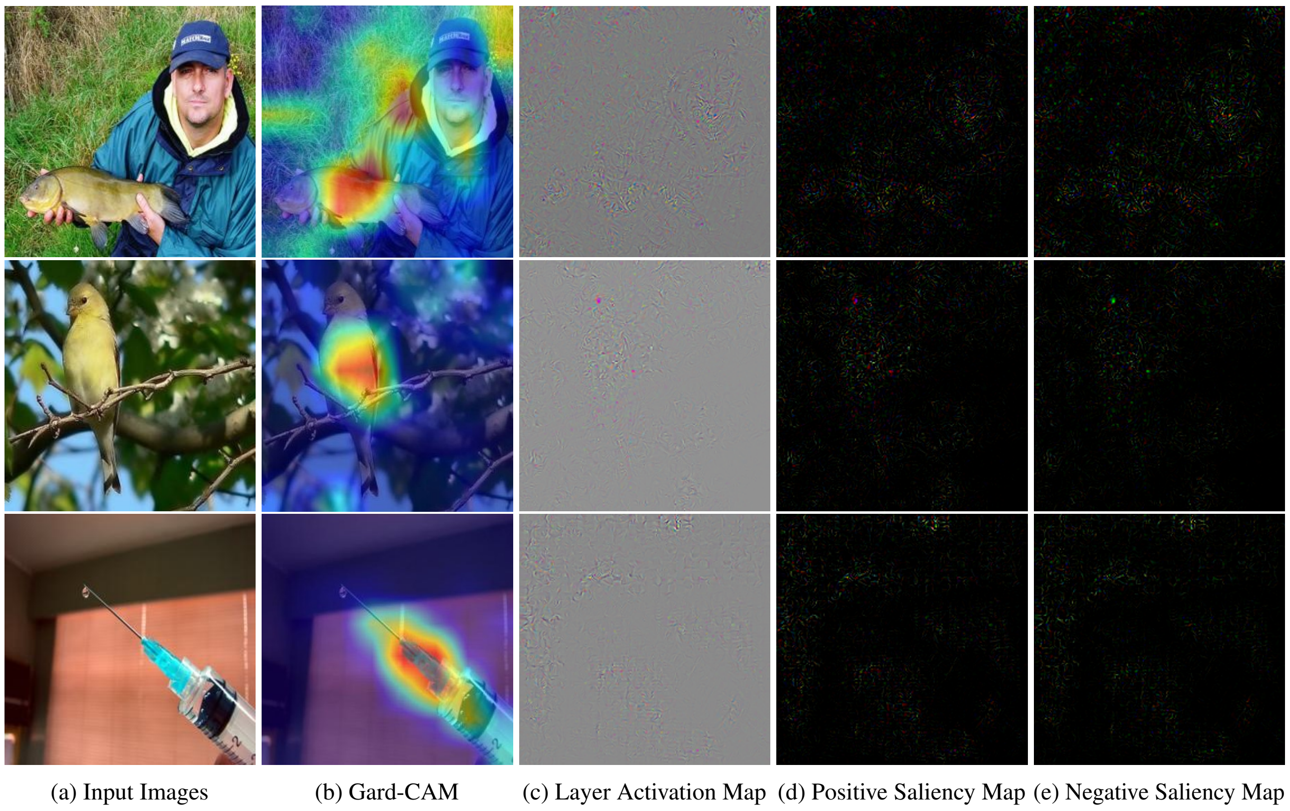
(a) Input Images (b) Gard-CAM (c) Layer Activation Map (d) Positive Saliency Map (e) Negative Saliency Map

Figure 4. Visualization results of Real Images for ResNeXt. (c)(d)(e) results are obtain from **Layer4** (last convolution layer)

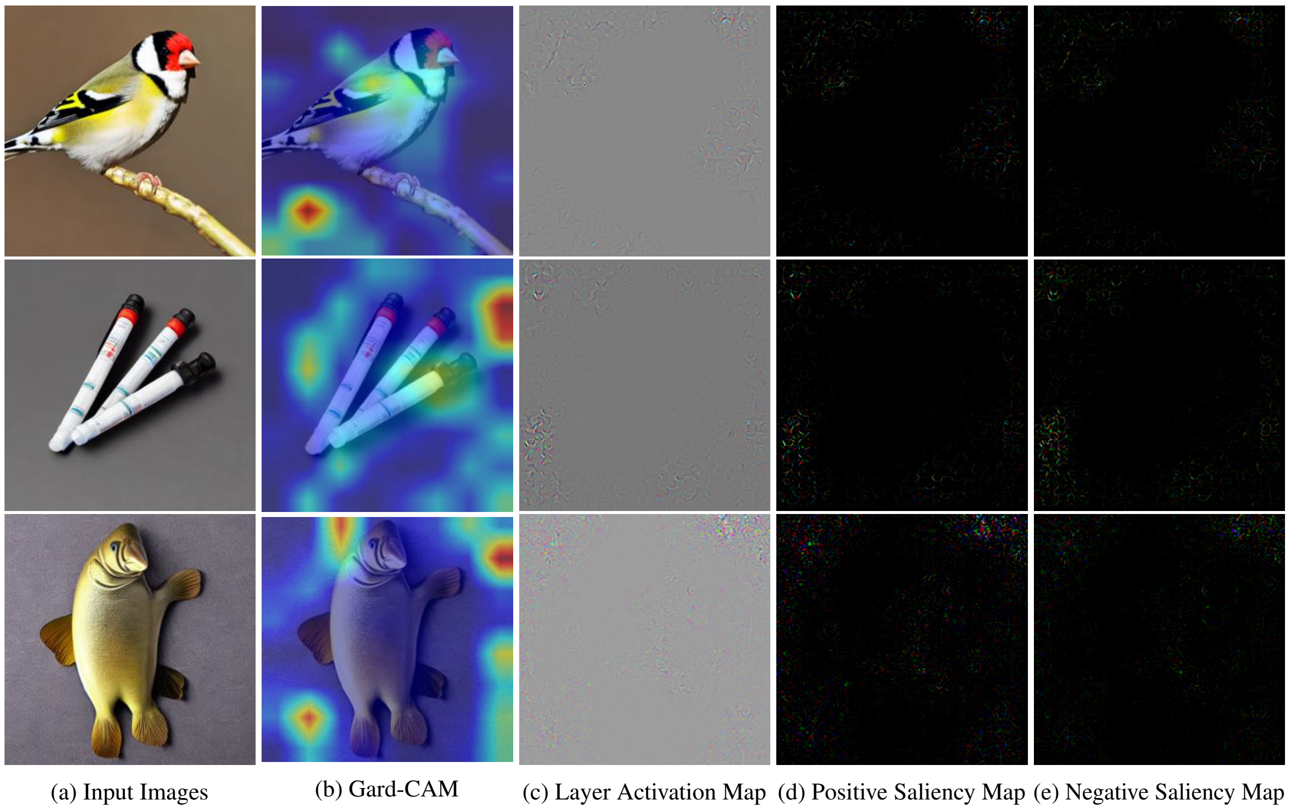
(a) Input Images (b) Gard-CAM (c) Layer Activation Map (d) Positive Saliency Map (e) Negative Saliency Map

Figure 5. Visualization results of Fake Images for ResNeXt. (c)(d)(e) results are obtain from **Layer4** (last convolution layer)

generated images, and there was a significant drop in performance, down to 50%-60% accuracy, which is almost equivalent to random guessing. However, by finetuning the training on the new dataset and using data augmentation techniques such as image blurring and image cropping, the model returned to almost perfect classification performance again.

5.2. Conclusions

In this study, we employed transfer learning on two top-performing models pretrained on ImageNet. While the fine-tuned ResNeXt101 model surpassed the fine-tuned SwinT model on the validation set, both models demonstrated equivalent performance on the test set. Subsequently, we augmented the ResNeXt101 model with additional layers and conducted transfer learning once more. The resulting model achieved 97.3% accuracy and a 97.3% F1 score on the test set. Given that our experiment utilized the entirety of Stable ImageNet-1k, we can assert with confidence that our model can reliably differentiate between synthetic and authentic images.

Moreover, we employed numerous prevalent convolution visualization techniques, such as Guided Backpropagation Saliency Maps and Grad-CAM, to identify the critical factors contributing to the classification of AI-generated content. Visualizations revealed that the image's background region serves as the primary determinant for classification. A majority of the results generated by the diffusion network featured smooth color backgrounds, which significantly influenced the classification process. The visualization outcomes further suggest the presence of a latent pattern in the background, rendering AI-generated content more easily detectable.

5.3. Limitations and future works

We have trained and tested our model on images with specific ratios and sizes, but we do not have sufficient evidence that the model could perform equally well on other images. In real-world applications, a synthetic image detector should be able to identify images based on some synthetic features, which are independent of their ratio and size.

As shown in our experiment heat maps, the model usually focuses on the background of the image. In a synthetic fish image, a human can confidently conclude it is a fake image, purely based on the observation that the fish has two tails, whereas the heat map shows that our model focused on the background, in other words, our model does not possess human common sense. A suggested future work could be finding a method that classifies the image not only based on its background but also the semantics of the image. We could also try and implement some more advanced feature extractions to further improve the accuracy of the model because our experiments have shown the relation between background texture and synthetic images.

Moreover, image synthesis is a fast involving area, new techniques like ControlNet (Zhang & Agrawala, 2023) emerged after we started this project, and there will be more efforts and achievements in the future. Identifying synthetic images will become harder, and we cannot stop pursuing approaches to mitigate the harm caused by AI.

References

Agarwal, Shruti, Farid, Hany, Gu, Yuming, He, Mingming, Nagano, Koki, and Li, Hao. Protecting world leaders against deep fakes. In *CVPR workshops*, volume 1, pp. 38, 2019.

Cetinic, Eva and She, James. Understanding and creating art with ai: Review and outlook. *ACM Trans. Multimedia Comput. Commun. Appl.*, 18(2), feb 2022. ISSN 1551-6857. doi: 10.1145/3475799. URL https://doi.org/10.1145/3475799.

Corvi, Riccardo, Cozzolino, Davide, Zingarini, Giada, Poggi, Giovanni, Nagano, Koki, and Verdoliva, Luisa. On the detection of synthetic images generated by diffusion models, 2022. URL https://arxiv.org/abs/2211.00680.

Elgammal, Ahmed M., Liu, Bingchen, Elhoseiny, Mohamed, and Mazzone, Marian. CAN: creative adversarial networks, generating "art" by learning about styles and deviating from style norms. *CoRR*, abs/1706.07068, 2017. URL http://arxiv.org/abs/1706.07068.

Gildenblat, Jacob and contributors. Pytorch library for cam methods. https://github.com/jacobgil/pytorch-grad-cam, 2021.

Gragnaniello, D., Cozzolino, D., Marra, F., Poggi, G., and Verdoliva, L. Are gan generated images easy to detect? a critical analysis of the state-of-the-art. In *2021 IEEE International Conference on Multimedia and Expo (ICME)*, pp. 1–6, 2021. doi: 10.1109/ICME51207.2021.9428429.

Kim, Hyeongwoo, Garrido, Pablo, Tewari, Ayush, Xu, Weipeng, Thies, Justus, Nießner, Matthias, Pérez, Patrick, Richardt, Christian, Zollhöfer, Michael, and Theobalt, Christian. Deep video portraits. *CoRR*, abs/1805.11714, 2018. URL http://arxiv.org/abs/1805.11714.

Krizhevsky, Alex, Sutskever, Ilya, and Hinton, Geoffrey E. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, 2017.

Lecun, Y., Bottou, L., Bengio, Y., and Haffner, P. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. doi: 10.1109/5.726791.

Li, Haodong, Li, Bin, Tan, Shunquan, and Huang, Jiwu. Detection of deep network generated images using disparities in color components. *CoRR*, abs/1808.07276, 2018. URL http://arxiv.org/abs/1808.07276.

Liu, Zhengzhe, Qi, Xiaojuan, Jia, Jiaya, and Torr, Philip H. S. Global texture enhancement for fake face detection in the wild. *CoRR*, abs/2002.00133, 2020. URL https://arxiv.org/abs/2002.00133.

Matern, Falko, Riess, Christian, and Stamminger, Marc. Exploiting visual artifacts to expose deepfakes and face manipulations. In *2019 IEEE Winter Applications of Computer Vision Workshops (WACVW)*, pp. 83–92, 2019. doi: 10.1109/WACVW.2019.00020.

Ozbulak, Utku. Pytorch convolutional neural network visualizations library. https://github.com/utkuozbulak/pytorch-cnn-visualizations, 2022.

Selvaraju, Ramprasaath R., Das, Abhishek, Vedantam, Ramakrishna, Cogswell, Michael, Parikh, Devi, and Batra, Dhruv. Grad-cam: Why did you say that? visual explanations from deep networks via gradient-based localization. *CoRR*, abs/1610.02391, 2016. URL http://arxiv.org/abs/1610.02391.

Sha, Zeyang, Li, Zheng, Yu, Ning, and Zhang, Yang. De-fake: Detection and attribution of fake images generated by text-to-image generation models, 2022. URL https://arxiv.org/abs/2210.06998.

Shorten, Connor and Khoshgoftaar, Taghi M. A survey on image data augmentation for deep learning. *Journal of big data*, 6(1):1–48, 2019.

Simonyan, Karen, Vedaldi, Andrea, and Zisserman, Andrew. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*, 2013.

Somepalli, Gowthami, Singla, Vasu, Goldblum, Micah, Geiping, Jonas, and Goldstein, Tom. Diffusion art or digital forgery? investigating data replication in diffusion models, 2022. URL https://arxiv.org/abs/2212.03860.

Suwajanakorn, Supasorn, Seitz, Steven M, and Kemelmacher-Shlizerman, Ira. Synthesizing obama: learning lip sync from audio. *ACM Transactions on Graphics (ToG)*, 36(4):1–13, 2017.

Svozil, Daniel, Kvasnicka, Vladimir, and Pospichal, Jiri. Introduction to multi-layer feed-forward neural networks. *Chemometrics and intelligent laboratory systems*, 39(1): 43–62, 1997.

Wang, Sheng-Yu, Wang, Oliver, Zhang, Richard, Owens, Andrew, and Efros, Alexei A. Cnn-generated images are surprisingly easy to spot... for now. *CoRR*, abs/1912.11035, 2019. URL http://arxiv.org/abs/1912.11035.

Zhang, Lvmin and Agrawala, Maneesh. Adding conditional control to text-to-image diffusion models. *arXiv preprint arXiv:2302.05543*, 2023.

Zhang, Xu, Karaman, Svebor, and Chang, Shih-Fu. Detecting and simulating artifacts in GAN fake images. *CoRR*, abs/1907.06515, 2019. URL http://arxiv.org/abs/1907.06515.