Few Shot Learning for Image Generation

Anish Batra
New York University
ab8166@nyu.edu

Prashant Mahajan
New York University
prm349@nyu.edu

Rohan Vartak
New York University
rpv224@nyu.edu

Abstract

Generative Adversarial Networks (GAN) boast impressive capacity to generate realistic images. However, like much of the field of deep learning, they require an inordinate amount of data to produce results, thereby limiting their usefulness in generating novelty. In the same vein, recent advances in meta-learning have opened the door to many few-shot learning applications. In the present work we explore 2 different methods for generating images from GANs using less number of base images. Our first approach is using the Reptile meta learning algorithm for a GAN, thereby generating new images from as little as 4 images of an unseen class. Our second approach is to experiment with different GAN architectures, augmenting the data and training the GANs and using transfer learning to generate new images with these GANs. We show which methods are the best for image generation in GANs using few shot learning.

1. Introduction

Deep Learning has proved to be a highly effective tool for artificial intelligence, especially in the field of Computer Vision. With deep learning techniques, our ability to understand and manipulate images has been exponentially increasing over the past few years. Though deep learning techniques are highly effective, one of the shortcomings is the inordinate amount of data required to produce results. Even state-of-the-art models require a few hundred images to produce good results. One of the solution for this problem is to generate more images of different classes.

Generative Adversarial Networks [1] (GAN) are a novel method of generating new images based on existing images, the new images being similar to existing images. GANs however require large amounts of data to generate images that are comprehensible and of good quality. Due to this the ability of GANs becomes restricted. Alternatively, if large amount of data is available for a particular class of images, it it easier to train a GAN. But the availability of abundant data provides little purpose for generating more data of the same type.

On the other hand, advances in Deep Learning like Meta-Learning Algorithms have helped to make the learning tasks much more efficient and allowed to train networks with significantly less amount of data. Popular meta learning algorithms include MAML [2] and Reptile [3] and are widely used in supervised and unsupervised learning. As these algorithms are general in their application, they are also applicable to few-shot image generation as demonstrated by Clouatre et al. [4]. In this project we aim to develop a deep learning system which will provide state-of-the-art performance for few shot image generation based on the work of Clouatre et al.

In summary, our main objectives are the following:

- Optimizing the DCGAN model meta-trained using Reptile algorithm.
- Trying new GAN architectures using transfer learning and data augmentation.

The concept can be used to train Convolutional Neural Network where we want to generate more images for a class with fewer images.

1.1. Related work

MAML is the most widely used algorithm for few shot learning. But it is not a good fit for training a GAN because it relies on the direction of the loss function to be linked with the quality of the model which may not hold true for GANs. Also it requires a test set for the evaluation of performance, which is not possible for a GAN.

The first successful attempt at few shot image generation was by Lake et al. [6], where the Omniglot [7] dataset was introduced. The Omniglot dataset contained both image and stroke data and was used to train a Bayesian model through Bayesian Program Learning. The pen strokes were represented as simple probabilistic programs and were hierarchically combined to generate images. The model could be trained on a single image of a previously unseen letter and generated binary images of the same letter.

Rezende et al. [8] used a sequential generative model, which used an attention module [9] to have Variantional

Auto Encoder [10] attend to a section of generated image sequentially. It trained on binary images from the Omniglot dataset to produce 28x28 and 52x52 sized images with few shot learning.

Bartunov et al. and Vetrov et al. [11] used matching networks which are memory-assisted networks that leverage external memory by employing attention to learn new concepts. This was also trained on the Omniglot dataset and generated binary images of 28x28.

Clouatre introduced the concept of applying Reptile algorithm for few shot image generation with GANs. Clouatre introduced a new dataset called FIGR-8 [4], which is intended to serve as a new bechmark for few shot image generation. FIGR dataset incorporates a wide variety of classes, with more complex classes representing real world images. It was able to generate images from unseen classes with as little as 4 samples on the MNIST and Omniglot datasets. This is achieved without any lenghty inference time, external memory and additonal data. Also there is no hyperparameter tuning.

1.2. Problem Formulation

We define this problem as a set of tasks T containing multiple task τ where each individual task τ is an image generation problem with one class if images X_τ and a loss L_τ. The aim is to find the, with meta-training, parameters Φ such that the loss L_τ is minimized with little data and little training. In essence, we want to:

$$minimize_\Phi \mathbb{E}_\tau[L_\tau(U_\tau^k(\Phi))] \tag{1}$$

where $U_\tau^k(\Phi)$ is the operator that updates Φ k times using x_n, a total of n data points sampled from X_τ .

MNIST As an example, the MNIST dataset contains 10 classes (the 10 digits). In the few-shot image generation problem, they represent 10 tasks to solve, τ_0 to τ_9. We choose τ_0 to τ_8 to be the training task and τ_9 to be the test task. Through meta-training on τ_0 to τ_8, we aim to obtain a set of parameters Φ that will quickly converge on a new τ. We choose n to be 4, meaning that we aim for our meta-trained Φ to converge to generating images of 9's with only 4 images sampled from τ_9.

2. Method

We have used the following methods to generate images.

- Transfer Learning and data augmentation with DCGAN, cDCGAN, InfoGAN.
- Meta-training on DCGAN with Reptile.

For the transfer learning and data augmentation method we have used the Fashion MNIST dataset, which contains images of common apparel and accessories like T-shirt, Trouser, Pullover, Dress, Coat, Sandal, Shirt, Sneaker, and Bag.

We have followed the following steps for training the GANs:

- Train the GAN on eight types of images which are T-shirt, Trouser, Pullover, Dress, Coat, Sandal, Shirt, Sneaker, and Bag of the FashionMNIST dataset for 20 epochs. Save the Generator's and Discriminator's learnable parameters in a file.
- Randomly choose four images of the last type Ankle Boot from the dataset and augment these images to 300 images using transformations like RandomCrop, ColorJitter etc.
- Load the Generator and Discriminator model with the weights stored in first step, and fine-tune it with images generated in the second step.

For the meta-learning approach we have trained our model on three different datasets which are MNIST, Omniglot and Fashion MNIST. We have kept the learning rate very small ie. 1e-3 to help the GAN train better. The learning rate of the inner loop was kept 1e-4. The images in each of the datasets are rescaled to 32 * 32 pixels. We used an Adam Optimizer with loss being Wasserstein loss with gradient penalty.

For the MNIST dataset the training classes were the images from 0 to 8. The test class was the digit 9. For the Onmiglot dataset the training classes were all the 1623 characters in the dataset minus 20 randomly sampled character classes for the test set. For Omniglot the batch size was changed to 8 for faster training while all the other parameters were kept the same as MNIST. For the Fashion MNIST dataset all the parameters were same as the MNIST dataset.

Reptile Algorithm works by repeatedly sampling a task, performing stochastic gradient descent on it and updating the initial parameters towards the final parameters learned on that task. Reptile is the application of the shortest descent algorithm in the meta-learning setting, and is mathematically similar to first order MAML that only needs black box access to an optimizer such as SGD or Adam, with similar computational efficiency and performance.

Like MAML, Reptile seeks an intialization for the parameters of a neural network, such that the network can be fine tuned on a small amount of data from a new task. While MAML unrolls and differentiates through the computation graph, Reptile simply performs SGD on each task in a standard way - it does not unroll a computation graph. This makes Reptile take less compuation and memory than MAML.

The adapted Reptile pseudo code for meta-training the model is depicted in Algorithm 1. The algorithm is composed of an outer loop and an inner loop. The inner loop is

the K step of the operator U on a copy of the parameters Φ with task τ. Once we have those adapted weight W_τ, we can proceed to the outer loop. We set the gradient of Φ to be equal to $\Phi - W_\tau$. We then take one step with the Adam optimizer [12].

Algorithm 1: FIGR training

```
Initialize Φ_d, the discriminator parameter vector
Initialize Φ_g, the generator parameter vector
for iteration 1, 2, 3 ... do
    Make a copy of Φ_d resulting in W_d
    Make a copy of Φ_g resulting in W_g
    Sample task τ
    Sample n images from X_τ resulting x_τ
    for K > 1 iterations do
        Generate latent vector z
        Generate fake images y with z and W_g
        Perform step of SGD update on W_d with
         Wasserstein GP loss and x_τ and y
        Generate latent vector z
        Perform step of SGD update on W_g with
         Wasserstein loss and z
    end for
    Set Φ_d gradient to be Φ_d - W_d
    Perform step of Adam update on Φ_d
    Set Φ_g gradient to be Φ_g - W_g
    Perform step of Adam update on Φ_g
end for
```

Once meta-trained, we use a similar process to generate novel images from the sampled class described in Algorithm 2.

Algorithm 2: FIGR generation

```
Using W_d, a copy of the meta-trained Φ_d
Using W_g, a copy of the meta-trained Φ_g
Sample test task τ
Sample n images as x_τ from X_τ
for K >= 1 iterations do
    Generate latent vector z
    Generate fake images y with z and W_g
    Perform step of SGD update on W_d with
     Wasserstein GP loss and x_τ and y
    Generate latent vector z
    Perform step of SGD update on W_g with
     Wasserstein loss and z
end for
Generate latent vector z
Generate fake images y
```

For every task τ there exist optimal discriminator and generator weights $W_{d\tau}$ and $W_{g\tau}$. Intuitively, Reptile initializes the weights Φ_d and Φ_g to the point in parameter space that minimizes the distance between Φ_d, Φ_g, $W_{d\tau}$ and $W_{g\tau}$ for all τ, or

$$minimize \sum_T (\Phi_d - W_{d\tau}) + (\Phi_g - W_{g\tau}) \qquad (2)$$

Hence, for a sampled task τ, a model optimized with Reptile can quickly and with few data points converge to the optimal point $W_{d\tau}$, $W_{g\tau}$ from Φ_d, Φ_g. If the test tasks are close enough to the training task and if the training tasks are numerous enough, Φ_d and Φ_g are likely to be close to a test τ's $W_{d\tau}$ and $W_{g\tau}$. This makes for rapid and easy generalization from few data points.

Reptile is broadly similar to joint training, and is effectively identical with a K of 1. However, by doing more gradient steps, we prioritize learning features that would be hard to reach, unlike joint training. Assuming a 2D parameter space, a K of 10 and a task τ; a local minimum for parameter 1, $W_{\tau 1}$, is reached after 2 gradient steps and a local minimum for the parameter 2, $W_{\tau 2}$, is not reached after K steps; it is probable that:

$$\Phi_1 - W_{\tau 1} < \Phi_2 - W_{\tau 2} \qquad (3)$$

This results in a larger outer loop update in the parameter space that is not readily attainable from Φ and smaller updates in the parameter space in which the model already possesses the ability to converge quickly.

3. Model Architecture

3.1. Meta-learning on DCGAN with Reptile

We used a DCGAN which was trained on the Wasserstein Loss [13] with gradient penaly [14]. In work by Clouatre et al. the model was trained for MNIST, Omniglot and FIGR-8 datasets. The generator and the discriminator of the GAN were built with residual neural networks [15] with 18 layers. The discriminator uses layer normalization based on work of Gulrajani et al. [14]. The generator also uses layer normalization since batch normalization's statistics are incompatible with Reptile's meta-update.

As described by Clouatre et al. the rectified units were Parametric ReLU [16] for a prelimnary model. All images were in grayscale and were resized to 32x32 or 64x64 and were normalized to have values between -1 and 1. We did not use data augmentation for this model.

For our prelimnary model we trained the model for 100,000 steps for the Omniglot dataset with the batch size 8. We are using 2 Tesla V100 GPUs on Google Could Platform. The Reptile algorithm decribed earlier is the main meta-learning algorithm for the model.

3.2. Transfer Learning and Data Augmentation using different GAN architectures

In addition to the above method we have also experimented using transfer learning and data augmentation on

different GAN architectures like CGAN [19] and InfoGAN [20]. The main idea here was to add labels as extension to the latent space to generate and discriminate images better.

In **CGAN** we added the labels of the images as a additional parameter to the generator. We also added the labels to discriminator input to distinguish real images better. The labels were 1-hot encoded. Below is a architecture of CGAN.

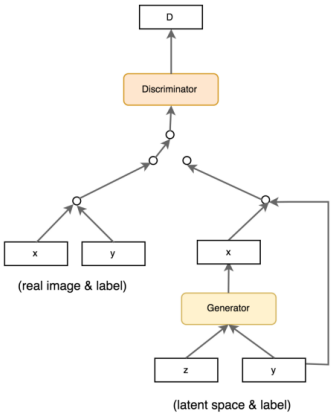

Figure 1: CGAN

InfoGAN works by minimizing mutual information. We pass a vector sampled from uniform distribution along with the latent space to the generator to generate images. The discriminator in InfoGAN outputs a probability distribution Q (c/x) along with the loss. InfoGAN works by subtracting an additional term from the loss function which is called as mutual information calculated using the Q. The mutual information equals to zero is the generated images and real image is different. If the real image and generated image is same then the mututal information term is high and ultimately the cost of the GAN is reduced.

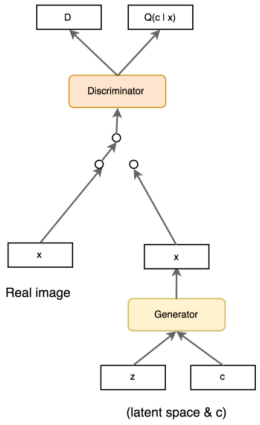

Figure 2: InfoGAN

4. Results

The results displayed below are for the different GAN architectures explained in the methodology. For each of the architecture we have supplied 300 images to the generator, through data augmentation. The generator trains on 300 images to generate new images. Each of the GAN below is run on the Fashion MNIST dataset.

Figure 3: DCGAN (Epoch 0, 5, 10 and 60)

The above are the results from DCGAN for the epochs 0, 5, 10 and 60. The ankel boot is the test class in this case.

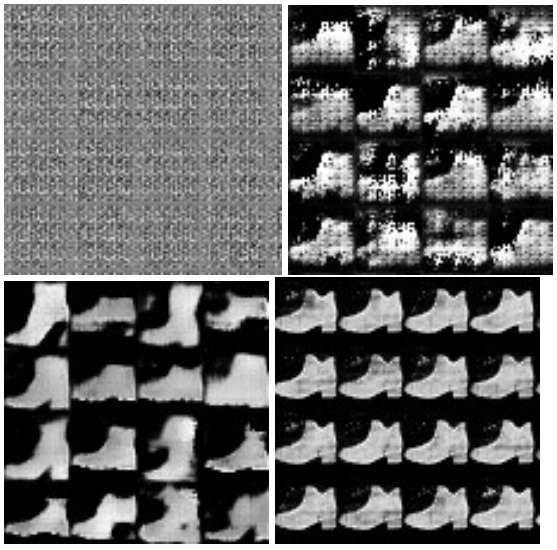

Figure 4: CGAN (Epoch 0, 20, 30 and 100)

Above are the results of CDCGAN for the epochs 0, 20, 30 and 100. Here too the ankel boot is the test class.

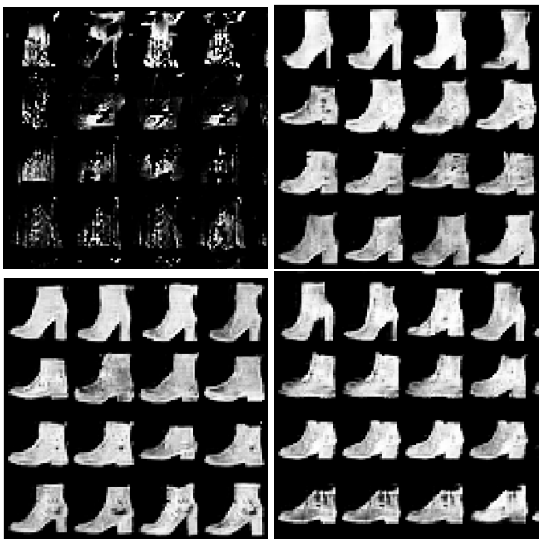

Figure 5: InfoGAN (Epoch 0, 6, 30 and 70)

Above we see the results of InfoGAN. The same test class of ankel boot is kept over here.

For the GAN using the reptile method we get the following results.

Figure 6: Reptile on MNIST

The above are the results for the Reptile algorithm on the MNIST dataset. The top 4 images are the images supplied to the GAN. The 12 images below are the ones generated by the GAN. Here the test class is the number 9.

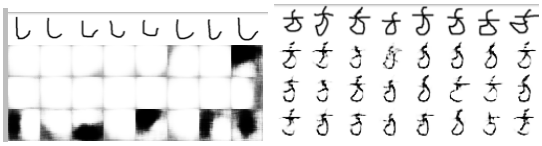

Figure 7: Reptile on Omniglot

The above are the results for the Reptile algorithm on the Omniglot dataset. Here 8 images are supplied to the GAN. The images below are the ones generated by the GAN. The test class is a pattern from the Omniglot dataset.

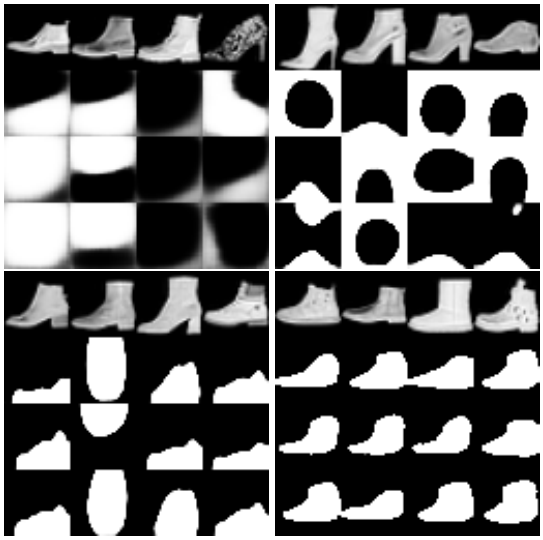

Figure 8: Reptile on Fashion MNIST

Above are the results of the Reptile algorithm on the Fashion MNIST data set for the epochs 1, 10000, 50000, and 100000. The top 4 images in each image are the input that is given to the GAN. Though we see that for epoch 100000 the results is not very good, but we can see that there is a considerable improvement in the images that are generated as compared to early epochs.

4.1. Evaluation

The results are evaluated using three criterion:

- Mean Square Error (MSE) - as the name suggests finds the mean square error between pixel intensities of the two images. One issue with this method is that large distances between pixel intensities do not necessarily mean the content of the images are dramatically different. In MSE value of 0 indicates the perfect similarity.
- Structural Similarity Index (SSIM) - attempts to model the perceived change in the structural information of the image. SSIM is used to compare two windows (i.e. small sub-samples) rather than the entire image as in MSE. Doing this leads to a more robust approach which accounts for changes in the structure of the image, rather than the perceived change. In SSIM, value can vary between -1 and 1, where 1 indicates perfect similarity.
- BRISQUE- stands for Blind/Referenceless Image Spatial Quality Evaluator which is majorly used in No-reference Image Quality Assessment (IQA). IQA aims to quantitatively represent the human perception of quality of the image. BRISQUE is a model that only uses the image pixels to calculate features. It relies on spatial Natural Scene Statistics (NSS) model of locally normalized luminance coefficients in the spatial

domain, as well as the model for pairwise products of these coefficients. The score used to represent image quality goes from 0 to 100. An image quality of 100 means that the image's quality is very bad. The results of the BRISQUE are justified considering the input images to the model generated the score of 71.

5. Conclusion

From the results we see that for the transfer learning method, we get good results using data augmentation where 300 images are sufficient enough to give quality images in less training steps. In contrast, for the Reptile algorithm, we get less satisfactory results because we are supplying only 4 images to the generator to generate more images. Still the generalizing ability of Reptile algorithm is higher and is in sense a better example of few shot learning because of the less number of images used by the generator.

6. Future Work

- Generating multichannel and larger images using datasets like CIFAR-100 and ImageNet.
- Experimenting with different GAN architectures for training with Reptile algorithm.
- Developing a pretrained GAN model trained on ImageNet dataset for PyTorch.

The code for implementation is available on Github at: https://github.com/Prashant-mahajan/Few-shot-Image-Generation

References

[1] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, DavidWarde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, Advances in Neural Information Processing Systems 27,*, pages 2672–2680. Curran Associates, Inc., 2014.

[2] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *CoRR, abs/1703.03400, 2017*

[3] Alex Nichol, Joshua Achiam, and John Schulman. On firstorder meta-learning algorithms, 2018.

[4] Clouâtre, L., Demers, M. (2019). FIGR: Few-shot Image Generation with Reptile. *arXiv preprint arXiv:1901.02199.*

[5] Deng, J. and Dong, W. and Socher, R. and Li, L.-J. and Li, K. and Fei-Fei, L. ImageNet: A Large-Scale Hierarchical Image Database. *http://www.image-net.org/papers/imagenet$_cvpr09.bib$*

[6] Brenden M. Lake, Ruslan Salakhutdinov, Jason Gross, and Joshua B. Tenenbaum. One shot learning of simple visual concepts.

[7] Lake, B. M., Salakhutdinov, R., and Tenenbaum, J. B. (2015). Human-level concept learning through probabilistic program induction. *Science, 350(6266), 1332-1338.*

[8] Danilo Rezende, Shakir, Ivo Danihelka, Karol Gregor, and Daan Wierstra. One-shot generalization in deep generative models. In Maria Florina Balcan and Kilian Q. Weinberger, editors *Proceedings of The 33rd International Conference on Machine Learning, volume 48 of Proceedings of Machine Learning Research, pages 1521–1529* New York, New York, USA, 20–22 Jun 2016. PMLR.

[9] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv e-prints, abs/1409.0473, September 2014.*

[10] Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. *CoRR, abs/1312.6114, 2013.*

[11] Sergey Bartunov and Dmitry P. Vetrov. Few-shot generative modelling with generative matching networks. *In AISTATS, 2018.*

[12] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2014.

[13] Martin Arjovsky, Soumith Chintala, and L´eon Bottou. Wasserstein generative adversarial networks. *In Doina Precup and Yee Whye Teh, editors, Proceedings of the 34th International Conference on Machine Learning, volume 70 of Proceedings of Machine Learning Research, pages 214–223.* International Convention Centre, Sydney, Australia, 06–11 Aug 2017. PMLR.

[14] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron Courville. Improved training of wasserstein gans, 2017..

[15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Jun 2016.*

[16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. *2015 IEEE International Conference on Computer Vision (ICCV), Dec 2015.*

[17] Xu, Q., Huang, G., Yuan, Y., Guo, C., Sun, Y., Wu, F., Weinberger, K. (2018). An empirical study on evaluation metrics of generative adversarial networks. *arXiv preprint arXiv:1806.07755.*

[18] Zhang, H., Xu, T., Li, H., Zhang, S., Wang, X., Huang, X., Metaxas, D. N. (2017). Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks. *In Proceedings of the IEEE International Conference on Computer Vision (pp. 5907-5915).*

[19] Mirza, M., Osindero, S. (2014). Conditional generative adversarial nets.

[20] Chen, X., Duan, Y., Houthooft, R., Schulman, J., Sutskever, I., Abbeel, P. (2016). Infogan: Interpretable representation learning by information maximizing generative adversarial nets. *In Advances in neural information processing systems (pp. 2172-2180).*

Algorithm	MSE	SSIM	BRISQUE
Reptile - MNIST	4104.78	0.73	-
Reptile - Omniglot	4320.69	0.91	-
Reptile - Fashion MNIST	10877.69	0.6	80.34
DCGAN - Fashion MNIST	3689.11	0.84	77.432
CDCGAN - Fashion MNIST	3436.85	0.8	77.1049
InfoGAN - Fashion MNIST	3863.17	0.75	74.67

Table 1: Evaluation of results