Domain Independence Analysis

Sai Ritesh Thela[1]

University of Florida, Gainesville Fl 32611, USA

Abstract. Semantic segmentation algorithms that utilize convolutional neural networks (CNNs) often require custom-labeled datasets, which can be time-consuming and labor-intensive to create. Additionally, these algorithms frequently underperform when dealing with out-of-distribution data. In this study, we aim to tackle these challenges by training an auto-encoder model on synthetic data and examining its transfer learning performance on Pascal VOC dataset.
To achieve this, we generate custom shape data and train our models using this information. Next, we assess the transfer learning capabilities of these auto-encoder models by applying them to the task of semantic segmentation of objects within the Pascal VOC dataset [1]. Our goal is to evaluate the generalizability of auto-encoder models to out-of-distribution data. Through this process, we aspire to discover novel strategies for enhancing the accuracy and efficiency of semantic segmentation in various applications.
By focusing on synthetic data and transfer learning, we hope to alleviate the need for extensive manual labeling efforts and improve algorithm performance on previously unseen data. This research has the potential to significantly contribute to the advancement of semantic segmentation techniques and their applications in diverse domains. Our findings may ultimately lead to more precise and efficient segmentation solutions.

Keywords: Semantic segmentation · Transfer Learning · Out-of-Distribution Data · Generalizability.

1 Introduction

The growing success of deep learning techniques in the field of computer vision can be attributed to the development of powerful models, as well as the availability of large-scale labeled datasets. These datasets have enabled the training of robust models for various tasks, such as object recognition, segmentation, and scene understanding. However, obtaining labeled data for specific tasks can be both labor-intensive and costly, particularly when it comes to tasks requiring fine-grained annotations or when dealing with less common categories or scenarios. This limitation has led researchers to explore alternative ways of obtaining training data, with synthetic data generation emerging as a promising avenue.

Synthetic data, generated through computational models, offers the advantage of being abundant, diverse, and customizable, which can significantly reduce the need for manual annotations. Furthermore, synthetic data can be utilized

to augment real-world datasets, providing additional training samples that can help improve the performance of deep learning models. In this project, our primary goal is to generate synthetic data in the form of complex visual patterns using noise-based textures and random shapes and to train a neural network on this data. Subsequently, we aim to test the transfer learning performance of the trained network on the Pascal VOC dataset, a widely-used benchmark for object recognition and segmentation tasks. By doing so, we seek to demonstrate the effectiveness of our approach in learning meaningful representations from synthetic data that can be transferred to real-world tasks, thus addressing the challenges associated with obtaining large-scale labeled data.

Our method for generating complex visual patterns combines multiple noise algorithms, such as Perlin Noise, Simplex Noise, Cellular Noise, to create diverse and intricate textures. These algorithms have been widely used in computer graphics and procedural content generation due to their ability to produce visually appealing and natural-looking patterns. In addition to these textures, we generate random shapes by sampling points and using Bézier curves, a powerful mathematical tool for creating smooth curves and shapes. We further enhance the complexity of our synthetic data by combining two or more shapes using the XOR operation, which introduces holes and increases the intricacy of the generated patterns.

To ensure that our synthetic images resemble real-world conditions, we place the generated shapes and textures on an image of size 224x224, and create a corresponding mask for each image. We also add noise to the images, emulating the variations and imperfections typically found in real-world data. This added complexity helps ensure that our trained neural network can generalize better to real-world tasks.

2 Synthetic Data set Generation

2.1 Random Texture Generation

Perlin Noise Perlin Noise [2] is a gradient noise algorithm developed by Ken Perlin in 1983. It has become widely-used in computer graphics, procedural content generation, and computer vision for generating visually appealing and natural-looking patterns, such as textures and terrains. The key advantage of Perlin noise lies in its smoothness and pseudo-randomness, which provide coherent patterns without noticeable regularities or abrupt changes.Perlin Noise is very useful as it can simulate various naturally occurring backgrounds such as

- Terrain and landscapes: Perlin noise can be used to generate realistic heightmaps for terrain and landscapes in computer graphics and video games. By adjusting the frequency, amplitude, and octaves, it is possible to create varied terrain features such as hills, valleys, mountains, and plains.
- Clouds and atmospheric effects: Perlin noise can be used to simulate cloud patterns, fog, and other atmospheric effects. By creating 2D or 3D noise patterns with varying frequencies and amplitudes, it can generate visually

appealing and natural-looking cloud formations and atmospheric phenomena.

Simplex Noise Simplex Noise [4] is a gradient noise algorithm developed by Ken Perlin as a powerful tool for generating visually appealing and natural-looking patterns in various applications, such as computer graphics, procedural content generation, and computer vision. These patterns include textures, terrains, and other complex structures that mimic real-world phenomena.

Generating data using Simplex Noise involves initializing the dimensions and resolution of the grid. In this algorithm, grid cells are formed by simplexes, which are equilateral triangles in 2D and tetrahedra in 3D. Each grid cell is assigned a random gradient vector, which can be uniformly distributed to ensure isotropy.

For each point in the space where a noise value needs to be calculated, the corresponding grid cell is determined, and the dot product of the gradient vector and the displacement vector is computed for each corner of the cell. This process generates a set of values representing the influence of each corner on the point.Simplex Noise is very useful as it can simulate various naturally occurring background such as

- Ocean waves and water surfaces: Simplex Noise can be utilized to generate realistic and dynamic water surfaces, such as ocean waves, ripples, and turbulence. By adjusting the frequency, amplitude, and octaves, it is possible to create a variety of waveforms and water effects suitable for different scenarios.
- Fire and flame patterns: Simplex Noise can be used to generate realistic fire and flame patterns for visual effects in computer graphics, animations, and video games. By adjusting the noise parameters and combining them with particle systems, it is possible to create dynamic and natural-looking fire simulations that mimic the behavior of real flames.

Cellular Noise Cellular Noise[5] also known as Worley Noise or Voronoi Noise, is a noise function that is widely used in computer graphics, procedural content generation, and computer vision. It is particularly useful for generating visually appealing and natural-looking patterns such as textures, terrains, and various intricate structures.

In the data generation process of Cellular Noise, dimensions and resolution of the space are defined where the noise will be generated. A set of seed points are randomly distributed across this space. The number and distribution of seed points can be adjusted according to the desired level of detail and pattern complexity.

For each point in the defined space, the distance to the nearest seed point is calculated. This distance value becomes the output noise value for that point. The distance metric used can be Euclidean, Manhattan, or any other suitable metric depending on the desired pattern characteristics.

To create more complex patterns, the distances to the nearest 'k' seed points can be combined or modified using various functions. For example, the distance values can be added, subtracted, or multiplied to generate different effects.

By adjusting the parameters such as the number of seed points, distance metric, and the combination functions, a wide range of natural-looking patterns can be generated using Cellular Noise. These patterns can resemble cracked surfaces, cellular structures, or irregular grid-like patterns, making it suitable for a variety of applications in computer graphics, animation, and procedural generation.Cellular Noise is very useful as it can simulate various naturally occurring background such as

- Cellular structures: Cellular Noise can generate organic patterns resembling cellular structures found in plants, animals, and other living organisms. These structures can be used to create biological textures, such as plant cells, skin, or microscopic organisms.
- Mineral patterns: Cellular Noise can be employed to simulate the appearance of mineral deposits, crystalline structures, and other geological formations. This can be particularly useful in creating realistic textures for rocks, minerals, and gemstones.

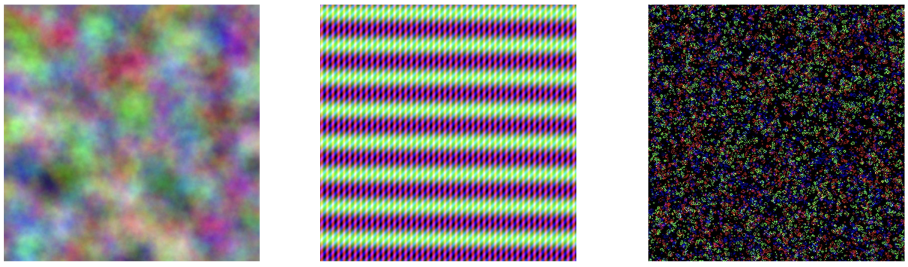

Fig. 1. Perlin Noise, Simplex Noise and Cellular Noise

2.2 Random Shape Generation

Simply-Connected Shapes Bezier curves [6], initially developed by French engineer Pierre Bézier while working at Renault, are smooth, parametric curves that are defined by a set of control points. These control points determine the shape and direction of the curve, making them a versatile tool in computer graphics, animation, and vector-based drawing applications for creating smooth, visually pleasing shapes and paths.

Generating random shapes with Bezier curves can be achieved by first defining a set of random points within a given space or image. These points will serve as the vertices of the shape you intend to create. By connecting these points

using Bezier curves, you form a closed and continuous path that outlines the resulting shape. The positions of the control points can be manipulated to create a diverse range of shapes, each with varying levels of complexity and smooth contours.

Adjusting the number of random points, the distribution of points within the space, and the positions of the control points allows for the generation of a wide variety of unique shapes. The smoothness of the resulting shapes is particularly beneficial for applications that require organic or aesthetically appealing forms.

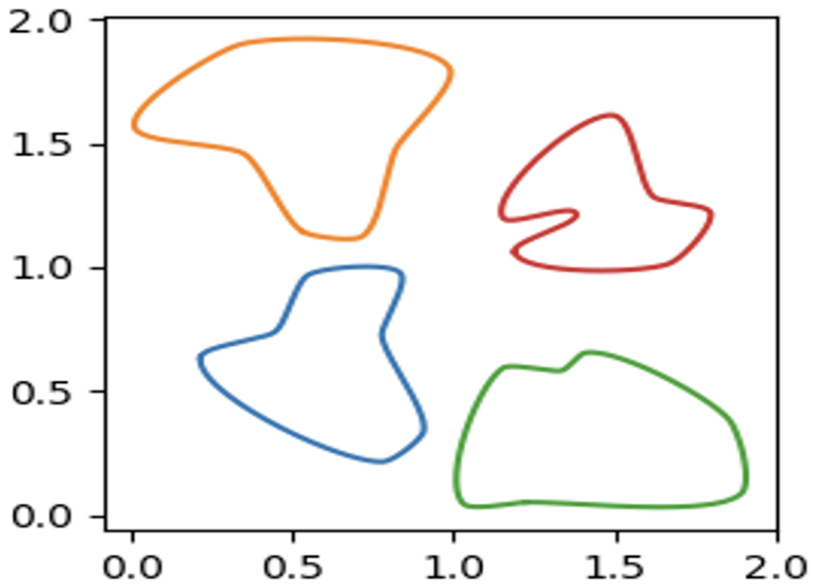

Fig. 2. Simply-Connected Shape Contour's

Complex Shapes The method of generating random shapes using Bezier curves primarily provides the outline of the shapes. When these shapes are filled, they do not contain any holes, resulting in simply connected forms. To create more complex shapes, you can combine two or more of these filled shapes using the XOR (exclusive or) operation. This process allows for the generation of intricate shapes with varying levels of detail and potentially creates shapes with holes or more complex boundaries. The combination of shapes through the XOR operation expands the possibilities for generating diverse and visually appealing shapes, suitable for a wide range of applications in computer graphics, animation, and procedural content generation.

Fig. 3. Complex Shapes

2.3 Final Dataset

To create the final dataset, a random texture is selected, and various transformations are applied to it. These transformations include scaling, shifting, and skewing the texture randomly, which adds diversity and complexity to the dataset. By applying these random transformations, you ensure that each instance of the texture in the dataset is unique and provides a wider range of variations for training or analysis.

Along with the transformed texture, a corresponding mask is also generated. This mask represents the regions where the texture is applied, helping to separate the texture from the background or other elements in the dataset. The combination of the transformed textures and their corresponding masks allows for the creation of a rich dataset

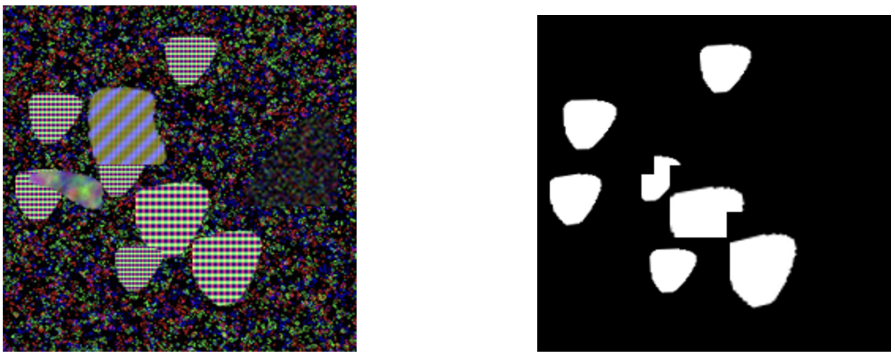

Fig. 4. Final Synthetic Image with Mask

2.4 Neural Network Architecture

Autoencoders [7] are valuable for semantic segmentation tasks because they can learn to capture and represent the underlying structure and features of input data. In the given Autoencoder class, which is designed to process 224x224 pixel images, the encoder and decoder components work together to compress and reconstruct the input images.

The encoder consists of two convolutional layers that progressively reduce the spatial dimensions of the input while capturing relevant features. The first layer takes an input with three channels (RGB) and outputs 64 feature maps, while the second layer takes these 64 feature maps and outputs 128 feature maps. Both layers use a kernel size of 3, stride of 2, and padding of 1.

The decoder part of the autoencoder reconstructs the original input data from the compressed latent representation using two transposed convolutional layers. The first layer upscales the 128 feature maps from the encoder to 64 feature maps, and the second layer upscales these 64 feature maps back to the original three channels (RGB). ReLU activation functions are used throughout the autoencoder for non-linearity, while a Sigmoid activation function is applied at the end of the decoder to ensure output values are in the range of 0 to 1, suitable for representing pixel values in an image.

This autoencoder architecture, designed for 224x224 input images, effectively learns and reconstructs the input data's structure and features, making it a suitable choice for tasks such as semantic segmentation.

2.5 Transfer Learning Dataset

The PASCAL VOC dataset is a popular benchmark in computer vision, particularly for semantic segmentation. It provides pixel-level annotations for diverse objects and scenes, enabling accurate model training and evaluation. The dataset's real-world challenges, such as varying scales and occlusions, promote the development of robust models. Its established reputation and community support make it an excellent choice for semantic segmentation tasks.

2.6 Experiments and Results

In the Experiments and Results subsection, we present the details of our autoencoder model's training process and its performance evaluation. The model was trained on a synthetic dataset consisting of 2,400 images, utilizing a batch size of 512 and spanning 300 epochs. During this process, we closely monitored the convergence rate, potential overfitting, and the impact of data augmentation techniques on the model's performance. Due to the complexity of analytically verifying the results, we compared the average starting loss and visually inspected the segmented images for quality assessment. Following the training phase, we evaluated the model's capabilities on both the synthetic dataset and the PASCAL VOC dataset. Through the use of visualizations such as loss curves and segmentation result examples, This analysis revealed the potential benefits

of employing synthetic data for semantic segmentation tasks and offered valuable insights for future research in this area.

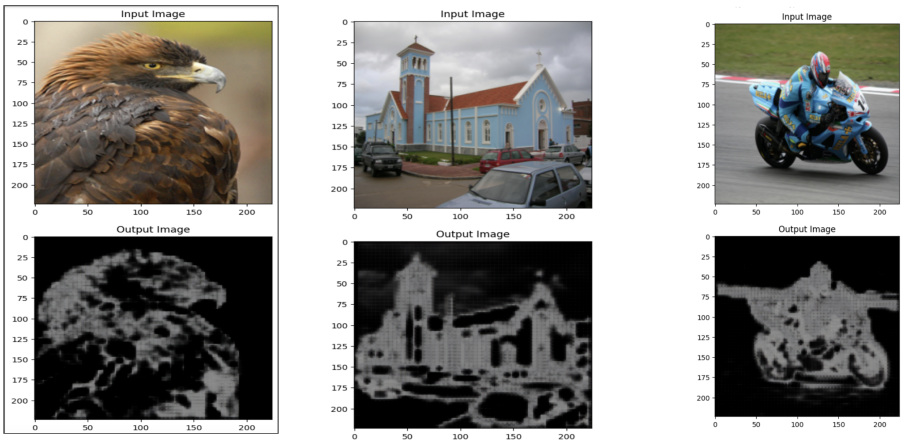

Fig. 5. Sample output Images

Table 1. Table displaying average initial loss with and without pre-training on the PASCAL VOC dataset.

	Loss without Pre-Training	Loss with Pre-Tranining
Avg Loss	0.13	0.05

3 Conclusion and Future Scope

In conclusion, our research demonstrates the potential of using synthetic data for training semantic segmentation models. The key insight is that there is a general concept of shape that goes beyond specific instances. By capturing the essence of shape, rather than fitting the model to specific examples, we can potentially enhance segmentation performance for a wide variety of shapes, even those that are out-of-distribution. A promising avenue for future work is to explore methods for conditioning models on shape information, which could lead to more robust and versatile segmentation models. Developing a way to condition models on the required shape would allow us to segment any shape, thereby advancing the field of semantic segmentation.

References

1. Everingham, M., Van Gool, L., Williams, C. K. I., Winn, J., Zisserman, A.: The PASCAL Visual Object Classes (VOC) Challenge. In: International Journal of Computer Vision, vol. 88, no. 2, pp. 303–338. Springer, New York (2010)
2. Perlin, K.: An image synthesizer. In: SIGGRAPH '85: Proceedings of the 12th Annual Conference on Computer Graphics and Interactive Techniques, pp. 287–296. ACM, New York (1985)
3. Perlin, K.: Improving noise. In: SIGGRAPH 2002: Proceedings of the 29th Annual Conference on Computer Graphics and Interactive Techniques, pp. 681–682. ACM, New York (2002)
4. Gustavson, S.: Simplex noise demystified. In: Technical Report, Linköping University, Sweden (2005)
5. Worley, S.: A cellular texture basis function. In: SIGGRAPH '96: Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques, pp. 291–294. ACM, New York (1996)
6. Bézier, P.: The mathematical basis of the UNISURF CAD system. In: Automotive Engineering Congress, Paper 710332. Society of Automotive Engineers, Detroit (1971)
7. Bengio, Y., Lamblin, P., Popovici, D., Larochelle, H.: Greedy layer-wise training of deep networks. In: Advances in Neural Information Processing Systems 19 (NIPS 2006), pp. 153–160. MIT Press, Cambridge (2007)