Synthetic Data-driven Prediction of Height for Childhood Malnutrition

David Berthiaume [*] [1] **Yuan Tang** [*] [1] **Chau Nguyen** [*] [1] **Catherine Gai** [*] [1] **Emilia Mazzolenis** [1] **Weiwei Pan** [1]

Abstract

While computer vision approaches have demonstrated success in various image-based tasks, they face challenges with early childhood height prediction for malnutrition detection due to a scarcity of publicly available training data. However, building public datasets for training and benchmarking machine learning models for this task is difficult because of the sensitive nature of the images. Although synthetic data have been employed in other data-scarce machine learning tasks, they do not exist for predicting children's height. In this work, we develop a novel generative algorithm to create synthetic images (including depth maps, segmentation maps, and keypoints) with *non-photorealistic human figures*, thereby providing an *ethical* and *scalable* solution to pre-train and evaluate computer vision models in a controlled setting. Our synthetic dataset models a wide variety of key real-world variables such as physical proportions, lighting, and posture. We demonstrate the potential of our dataset in a transfer learning setting by showing that models pre-trained on our synthetic data outperform baseline approaches when applied to real-world prediction tasks.

1. Introduction

Despite recent progress, malnutrition remains a significant global health problem, especially among children under the age of two in developing countries. Early detection and intervention are extremely important in preventing adverse health consequences for these children and promoting their future healthy growth and development (Hasegawa et al., 2017). Height is a key factor in determining if a child is malnourished, so being able to measure this accurately is critical (Ezzat et al., 2022). Traditional methods for obtaining heights suffer from two challenges: (1) manual height measurements with measuring tape can be prone to measurement error (Mikula et al., 2016), especially for children under the age of two due to their inability to keep a fixed posture; (2) manual measurement methods required specially trained personnel, disadvantaging resource-poor regions (Lazzerini & Lassi, 2019). Given the profound implications of inaccurate height measurements for estimating children's malnutrition, developing more accurate and efficient methods for measuring children's height is crucial.

To overcome these challenges, we wish to use machine learning to automatically predict heights from pictures of children taken by cell phones. Since cell phones are highly portable and widely available, this would make measuring height more widely accessible.

While deep learning techniques have been remarkably successful in various image-based tasks, obtaining data for training deep learning models can be challenging. The data must be labeled and annotated, which can be time-consuming and expensive. Obtaining images of subjects for childhood malnutrition prediction is even more challenging. These images and height measurements must be taken in a controlled environment by trained personnel to ensure high accuracy, and privacy is a significant concern.

Synthetic data can help alleviate some of these challenges. When implemented effectively, it allows for the training of computer vision models in a safe and ethical manner while still capturing important visual factors and real-world variations found in real images. Combining synthetic data with transfer learning could improve model generalizability and reduce the size of the real-world dataset needed.

We present a novel method to generate synthetic datasets for pre-training and benchmarking computer visions models in the task of early-childhood malnutrition detection. Our synthetic dataset is designed to satisfy the above two important criteria. Our contributions are as follows:

1. We present an algorithm to create synthetic images, depth maps, segmentation maps, and keypoints that capture key variations in real images (e.g. proportions,

[*]Equal contribution [1]School of Engineering and Applied Sciences, Harvard University, Cambridge, MA, USA. Correspondence to: David Berthiaume <dberthiaume@g.harvard.edu>, Yuan Tang <yuantang@g.harvard.edu>.

Proceedings of the 38th International Conference on Machine Learning, PMLR 139, 2021. Copyright 2021 by the author(s).

posture, light, and texture).

2. We provide a safe, ethical testing ground for researchers to develop and pre-train models prior to fine-tuning on limited real-data.

We benchmark a range of traditional computer vision models, such as convolutional neural networks (CNNs) and Vision Transformers (ViTs), on the synthetic dataset to explore height prediction performance across different neural network architectures. We demonstrate the promise of our dataset in a transfer learning setting.

Furthermore, we compare the performance of several computer vision models with and without pretaining on our synthetic dataset to show that our synthetic dataset provides a performance boost. This enhancement is especially evident where real labeled images are scarce or difficult to obtain. We show that pre-training models on our synthetic data leads to better transfer to a real prediction task, when compared with baseline approaches.

2. Related Works

2.1. Modeling

2.1.1. Height Prediction from Images

Predicting anthropometric measurements, particularly height, from images has been an active area of research. Early works employed single-image methods, which typically require a reference object of known dimensions for calibration, leading to relatively high errors of around 5.5% (Lee, 2012). Researchers have also explored Bayesian-like frameworks that incorporate prior knowledge of human anthropometry to eliminate the need for reference objects (Ben-Abdelkader & Yacoob, 2008). Multiple-view approaches have been proposed, enabling 3D reconstruction of the body for improved height estimation with an error of around 1% (Li et al., 2012). Depth-based techniques have enhanced accuracy to within 0.9% error by leveraging additional depth information (Yin & Zhou, 2020). However, most of these works focused on adult subjects rather than young children.

Convolutional Neural Networks (CNNs) have been widely used for height prediction tasks, with models like MobileNet, VGG16, GoogleNet, and AlexNet achieving mean absolute errors (MAEs) below 1.1 cm for height estimation in infants (Shu et al., 2023). These work have focused on using real-life, non-synthetic images dataset.

2.1.2. Multi-View and Fusion Techniques

To effectively utilize multiple views of an object, researchers have explored multi-view convolutional neural networks (MVCNNs) (Su et al., 2015). These models process each view through shared convolutional layers and aggregate the view-specific features using additional layers for the final prediction task, enabling information sharing across views.

Fusion-based methods (Ramirez et al., 2021) combine features or decisions from multiple views or image types, such as fusing RGB images with depth maps at different stages of the processing pipeline. Transformer models (Dosovitskiy et al., 2020), inspired by natural language processing, have also shown promise in handling dependencies between multiple views of an object for classification or regression tasks. However, these have not been applied in the context of height prediction for children in lying position.

2.1.3. Malnutrition Detection

In the context of malnutrition detection, researchers have investigated two main approaches (Rajappan et al., 2023). The first one involves directly classifying malnutrition status from images, with models like EfficientNet-B4 achieving 96% accuracy. The second approach predicts body mass index (BMI) from images and uses it to identify malnutrition, with models like ResNet34 yielding 92% accuracy.

2.2. Datasets for Height Prediction and Malnutrition Detection

2.2.1. Real-world data

Several datasets have been curated for height prediction tasks. Trivedi et al. (Trivedi et al., 2021) collected a dataset of depth images for children under five years old from rural India, aimed at height estimation. However, this dataset focused on standing children and did not specifically target other postures often encountered in the field such as children lying down on a mat. Yin and Zhou (Yin & Zhou, 2020) created datasets of RBG images and depth maps, but for adult subjects rather than children. There are significant differences between the anatomy and body length proportions of children and adults that may make it difficult to predict the heights of children based on training with this dataset alone, however. A more general dataset that includes the types of proportions and features expected to be present in children is needed.

2.2.2. Synthetic Data

To overcome the challenges of collecting real-world data, researchers have also explored synthetic data generation approaches. Peng et al. (Peng et al., 2018) surveyed and discussed the use of synthetic data for training computer vision deep learning models in general. Rhodin et al. (Rhodin et al., 2018) generated a synthetic dataset of 3D human poses for unsupervised representation learning. Varol et al. (Varol et al., 2017) presented synthetically-generated but realistic images of people rendered from 3D sequences of human motion capture data to train CNNs for human depth estimation and human part segmentation tasks. These synthetic datasets were beneficial for pose estimation and segmentation. They do not contain heights that can be used to train a computer vision model, however.

Lurch et al. (Lurch et al., 2020) augmented the data for 3D pose estimation with synthetically generated dataset covering a more continuous pose space than the existing one, demonstrating the potential of synthetic data in overcoming the scarcity of real-world datasets. Additionally, Kubric (Kubric, 2019) is a scalable dataset generator that can create synthetic images with varying scenes, objects, and annotations, enabling the exploration of various computer vision tasks ranging from studying 3D NeRF models to optical flow estimation. While these datasets demonstrate the utility of synthetic data in training deep learning models, neither of them contain heights for human subjects that can be used to train a computer vision model for height prediction.

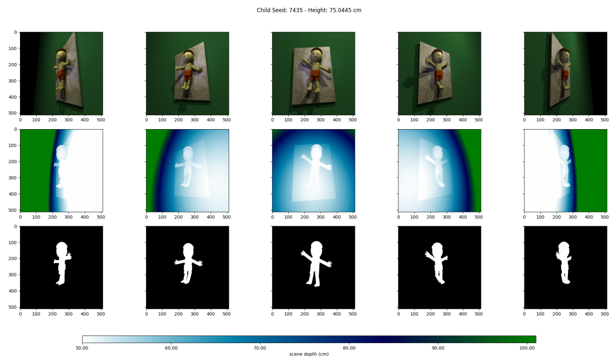

Figure 1. Example of a synthetic subject generated by FigureSynth. The upper row consists of color images, the middle row consists of depth maps, and the bottom row consists of binary segmentation maps indicating whether each pixel is part of the subject or part of the background.

3. Background

In the subsequent section, we discuss a pipeline for generating synthetic data for use in estimating a subject's height using computer graphic techniques. The synthetic data comprises RGB images, depth images, and segmentation maps of children lying on mats in various positions, lighting conditions, and camera view angles. The following terminology and notation are used throughout:

RGB images: color images that represent the scene in the red, green, and blue color channels.

Depth images: grayscale images where each pixel represents the distance of that point in the scene from the camera.

Ray tracing: a technique used to generate synthetic images by tracing the path of light rays through a virtual scene and simulating their interactions with objects.

Procedural images: images generated through computer programs rather than captured from the real world.

Parameter space: the range of values that the various parameters used to generate the synthetic data can take.

Keypoints: specific locations on the synthetic subject, such as the top of the head, the center of the hips, and the hands and feet, that are automatically generated and labeled for use in computer vision approaches.

4. Methods

In this section, we describe FigureSynth, an algorithm that generates images of human postures that (1) have pixel-perfect height measurements and (2) capture key variations in real images of subjects being assessed for early childhood malnutrition. In Section 5.1, we show that the generated synthetic data can be used to evaluate and pre-train deep learning models for height prediction.

4.1. Overview of FigureSynth

FigureSynth is a command-line program that uses procedural modeling and ray tracing powered by Pov-RAY (pov) to generate color images, depth maps, segmentation maps, keypoints, precise height measurements, and auxiliary information for synthetic children lying on a mat as viewed from different camera angles. Figure 1 shows an example of a synthetic child generated by FigureSynth.

The goal of FigureSynth is to parametrically model foreground subjects with a wide range of variations and with pixel-perfect measurements. Every foreground element – including physical proportions, postures, and textures – is modeled parametrically and can be individually specified or sampled from various random distributions. Elements of the scene, including camera position, lighting, and image quality, are also parametrically modeled. One can specify over 100 different parameters to control how the foreground subjects and scenes are rendered.

We argue that FigureSynth provides two important benefits in addition to benchmarking the model for height prediction:

- **Model Explainability:** By providing precise measurements for over 100 different features of the generated image, this data provides a rich environment for model explainability. For example, by analyzing the metadata attached to each image, one may find that a particular model is performing poorly on synthetic images where subjects have a particular type of limb articulation. *These explanations can guide us in making targeted and effective improvements to our models.*

- **Generalization to Other Tasks:** Height prediction is just one of the many tasks that we can benchmark using our synthetic dataset. Any parameter such as body width, leg length, leg position, or even keypoint locations (explained below) can be used in a prediction task. In fact, we can decompose height prediction into a set of sub-tasks, for which we can train models using FigureSynth. Combining models for multiple sub-tasks can potentially improve overall height prediction.

We emphasize that, with FigureSynth, we specifically avoid creating photorealistic images. The generation or curation of photorealistic images of children for machine learning can be problematic from an ethical standpoint, even when the end goal is for the public good.

Using real images of children without proper consent can lead to significant ethical issues (Ghosh, 2023). Real images of children contain personal information that can be used to identify individuals in some cases, leading to privacy breaches. Protecting the privacy of children is critical since they are a vulnerable population. Furthermore, using real images can lead to stigmatization of the children, especially if these images are publicly shared or the children are identified as malnourished.

Even realistic-looking synthetic images, though they do not directly infringe on privacy, can still raise ethical questions about the implications of using such data for training models, especially if some of these images look like actual individuals, even if by chance. (Hansen, 2023)

Another significant ethical concern is that realistic-looking synthetic images can potentially be used to create fake or misleading content. These images could be used to spread misinformation or even harass individuals. An individual might create fake images of someone in compromising situations and distribute them online, leading to harm.

We note that photorealism is not always necessary for achieving good results in training computer vision models. Depending on the specific application and tasks, computer vision systems can be effectively trained using synthetic or stylized imagery that does not mimic the full complexity of real-world visuals but instead retains essential structural and contextual elements needed for the model to learn and perform accurately. For example, non-photorealistic rendering was successfully used to train models for autonomous driving in complex urban environments under many weather conditions, where logistical challenges prevent training and testing such systems in the physical world (Dosovitskiy et al., 2017). Non-photorealistic rendering was also successfully used by OpenAI for reinforcement learning applications in robotics (OpenAI, 2018). Furthermore, using realistic static meshes or generative AI would greatly reduce this flexibility and make it difficult to generate the vast number of images with precise and exhaustive measurements needed to train a deep-learning model. In Section 5.1, we demonstrate that models that are pre-trained on our non-photorealistic synthetic data can transfer well to real photorealistic datasets.

4.2. Parametrization of Foreground and Scene

FigureSynth renders each parametrically modeled scene using ray tracing. The geometries in the scene are composed of spheres, cylinders, cubes, rounded cubes, triangles, planes, and BLOBs (Binary Large OBjects), explained below.

Each synthetic subject is rendered from multiple view angles to provide a variety of perspectives. The camera is placed at different positions above the subject to capture the body from multiple positions. These multiple views allow one to predict the subject's height with multi-view computer vision approaches. FigureSynth renders a depth image corresponding to each color image for each view by storing the distance from the camera to the first solid ray intersection in the scene. This depth image and the color imagery are precisely co-registered.

Finally, FigureSynth renders a segmentation map for each subject. The segmentation map is a binary image where each pixel is equal to 1 if it is part of the subject and 0 if it is part of the background. Along with the color and depth images, the segmentation map is helpful for training deep-learning models to predict the subject's height.

The primary function of FigureSynth is to use pseudorandom numbers to render millions of synthetic images, with no two being identical. The program accomplishes this by sampling over 100 parameters from various probability distributions. For example, FigureSynth may place an optional spotlight around the subject as a light source. The position of this spotlight is sampled from several normal distributions, with the light being most likely placed near the subject. The color of this light source is sampled from three uniform distributions, one for each of the red, green, and blue intensities.

Several random distributions are available in FigureSynth, including Bernoulli, Uniform, Normal, and Binomial. All parameters can be optionally specified in the command line and support every probability distribution. Every parameter has a default distribution that is used when no distribution is explicitly provided.

Most of the geometry of the synthetic subjects is composed of BLOBs. BLOBs are a powerful tool for creating smooth shapes parametrically with ray tracing. BLOBs are created from one or more subcomponents, each typically defined by a field of influence. These fields influence the density of the medium at their respective locations. When combined, they create a smooth surface based on a density threshold.

The surface of the BLOB is defined wherever the combined density of all influence spheres exceeds this threshold.

For example, consider a simple BLOB composed of two spheres. Sphere 1 has center $c_1 = (0, 0, 0)$, radius $r_1 = 1$, and strength $s_1 = 1.5$. Sphere 2 has center $c_2 = (1, 0, 0)$, radius $r_2 = 1$, and strength $s_2 = 1.5$.

We define a threshold t_r that determines how spread and smooth the shape is. A lower threshold means a larger, more spread-out blob as the surface includes points with lower summed field strengths. A higher threshold creates a tighter, smaller blob as only points with higher field strengths are included.

The influence functions for each sphere are given by

$$f_1(p) = 1.5\left(1 - \frac{\|p - (0,0,0)\|^2}{1^2}\right)^2,$$

$$f_2(p) = 1.5\left(1 - \frac{\|p - (1,0,0)\|^2}{1^2}\right)^2.$$

The total strength at any point p is:

$$S(p) = f_1(p) + f_2(p).$$

The points, p, inside the BLOB are all p such that $S(p) \geq t_r$.

Blobs can consist of spheres and cylinders, and the strength and shape of each sphere and cylinder can be individually controlled to create a wide variety of shapes from the resulting field. These composite objects are beneficial for creating smooth shapes with fine-grained control over the individual features of the shape. Much of the complexity of the subject's body is modeled using BLOBs, especially for the feet and hands (see figure 2).

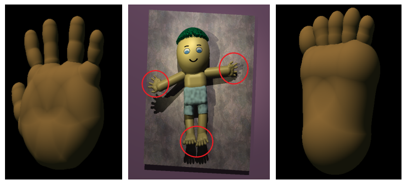

Figure 2. Hands and feet are rendered using BLOBs that consist of a complex arrangement of parametric spheres and cylinders. The strength and shape of each sphere and cylinder can be individually controlled to create a variety of shapes from the resulting field.

FigureSynth uses a hierarchical modeling approach to create the subject's body, with a central skeleton and parametric definitions for parent-child relationships. These parent-child relationships change dynamically as the different body parts are generated based on the samples from the random distributions. For example, the allowed upper arm rotations change based on the position and rotation of the main body to ensure that the arms are positioned realistically and that there is no clipping with the mat below.

FigureSynth defines every body part and object in the scene using a set of parameters and rules to assemble those parameters into geometry. For example, the upper leg is defined as a blob composed of a main cylinder and several spheres. These sub-components are generated based on the upper leg length, circumference, and starting and ending positions.

Synthetic subjects and their environments are rendered with a combination of static and procedural textures. These textures provide a degree of realism and variability for the models to learn from diverse scenes and improve their robustness in real-world applications. By simulating various textures, the models can be trained to better recognize and analyze patterns in different contexts, thereby enhancing their accuracy in tasks such as predicting the subject's height.

The material for the mat that the subject is lying on is sampled from 100 static public domain textures, which are available from `OpenGameArt.org` (Textures, 2024). These textures help provide considerable variety in the background of the images, which helps prevent the neural network from expecting a specific type of background. The textures are sampled using the choice distribution by default, which allows one to select one of the 100 textures with uniform probability.

Most of the textures in the scene are generated dynamically. Clothing typically has wrinkles and imperfections that are difficult to model procedurally. To create realistic clothing textures, FigureSynth uses a combination of procedural modeling, simplex noise, and normal maps. Having a perfectly smooth texture for the clothing would be unrealistic and cause the neural network to expect something that would not be present in real-world images. To simulate wrinkles in the clothing, a normal map is procedurally generated using simplex noise and then applied to the geometry.

A dynamic texture is created for the face of every subject to provide basic facial features. This texture is rendered from scratch using pycairo (Henstridge). The face texture is created by drawing a series of shapes, such as circles, rectangles, and lines, to create a face with eyes, a nose, and a mouth. The color of the face, by default, is sampled from a normal distribution to provide a wide variety of skin tones. Figure 3 shows four example face textures.

4.3. Lighting

By default, FigureSynth provides 4 light sources in each scene, 2 point lights, and 2 spotlights. The intensity of the lights is sampled from a uniform distribution to provide soft lighting for the scene with lights provided from multiple directions. Each of these lights can by controlled by tuning the intensity and their positions. Figure 4 shows an example of

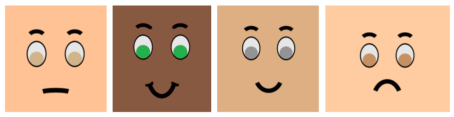
Figure 3. Example face textures rendered dynamically using pycairo. The color of the face is sampled from a normal distribution to provide a wide variety of skin tones and is chosen to closely match the skin color of the rest of the child's body.

a single subject rendered with various light configurations.

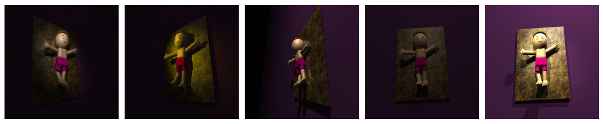
Figure 4. A single subject rendered with various light configurations. The intensity and position of each light source can be controlled to provide a wide variety of lighting conditions.

FigureSynth provides the ability to add noise to the rendered images to simulate real-world image noise. This noise is sampled from a Gaussian distribution with a mean of 0 and a standard deviation that can be controlled by the user. The noise is added to the color image and the depth map. Figure 5 shows an example of a subject rendered with varying levels of image noise. An additional noise type is provided to exclude small patches of the image and depth maps to simulate the effect of dropout in the neural network.

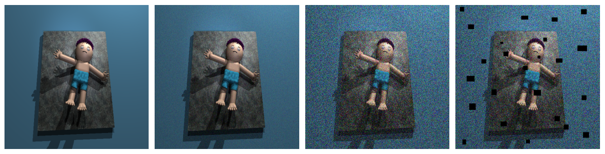
Figure 5. A single subject rendered with various noise levels. The noise is sampled from a Gaussian distribution with a mean of 0 and a standard deviation that can be controlled by the user. Optionally, box noise can be applied to the image.

FigureSynth provides a metadata file for each rendered image that contains the height of the subject along with dozens of other parameters that could be used for prediction tasks, including all of the random parameters used in the scene, the camera position, the lighting conditions, the image quality, and all keypoints, both the 3D coordinates and the 2D pixel coordinates. This metadata file is saved in text format. Appendix C shows an example of a portion of a metadata file or a single render. One could use this information to train a neural network to predict hair color or the distance of the camera from the child for example.

Many computer vision tasks require keypoint detection, such as object tracking, pose estimation, and facial recognition. FigureSynth provides the ability to generate pixel-perfect keypoint coordinates for each subject. The keypoints are generated based on the dynamic geometry of the subjects and are then projected onto image space and saved to the metadata file associated with each view.

5. Experiments

5.1. Experiment 1: Validating Data Generated from FigureSynth

In this set of experiments, we demonstrate that data generated from FigureSynth can be used for training height prediction models. Specifically, we hypothesize that the synthetic images we generate contain sufficient elements of realism to provide a reliable signal for predicting height. To test this hypothesis, we benchmark several state-of-the-art Computer Vision models for predicting subjects' height based on the input images and show that all models attain excellent performance.

We anticipate that state-of-the-art models will learn from synthetic data effectively, achieving high accuracy (95%+), a high R-Squared value (0.95+), and a low mean absolute error (less than 1). Additionally, we expect these advanced models to outperform our baseline model, which only has three convolutional layers. Given that deeper models and those incorporating self-attention mechanisms possess greater capacity to discern patterns, they should be more adept at predicting heights.

5.1.1. Set Up

We consider the following models:

- Vanilla 3-Layer Convolutional neural network (CNN)
- ResNet Family: ResNet-18, ResNet-34, ResNet-50, ResNet-101
- VGG Family: VGG-16 with Batch Normalization, VGG19 with Batch Normalization
- MobileNet Family: MobileNet-V3-Small, MobileNet-V3-Large
- Vision Transformer (ViT) Family: ViT-base-16, ViT-base-32

We set our Baseline model to be the 3-layer CNN model.

Except from ViT, all other models are variants of CNN. We aligned the input and output format among all networks, and developed a framework to streamline the whole process from model selection, training, validation, result visualization, and reporting test statistics. We demonstrated the generalizability of the framework by applying it universally to all models defined above.

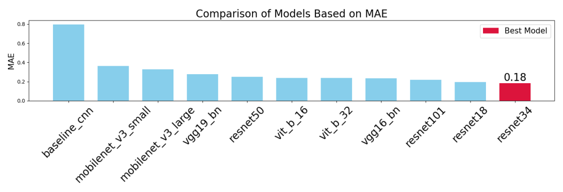

Figure 6. MAE Performance Comparison

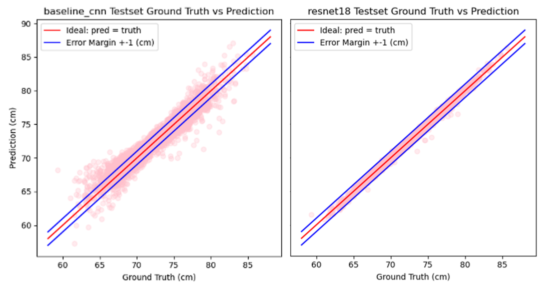

Figure 7. Ground Truth vs. Prediction: The left-hand side shows results for the baseline, and the right-hand side shows ResNet-18. Pink points represent individual samples from the test set. Points directly on the red line indicate perfect predictions. The pink points that fall within the boundaries marked by two blue lines are considered accurate predictions, with an error margin of ±1 cm.

For all the models, we modify their fully connected readout by applying the same linear layer sequence, with the last layer having one neuron, which is used to make height predictions. We trained the models with consistent hyper-parameters, with specific settings detailed in the appendix. We also apply consistent data augmentation on-the-fly across all model training process.

Inputs and Outputs The input to our models consists of depth maps with dimensions of $3 \times 224 \times 224$. To accommodate the input specifications of pre-trained models, we have channel-wise stacked the original single-channel depth map into three channels. This manipulation ensures compatibility with the architectures that typically expect three-channel RGB images. The output across all models is a scalar value, which stands for the predicted height.

Evaluation Metrics We report and compare model performance with 3 different metrics: Mean-Absolute Error, One-centimeter Tolerance Accuracy, and R-squared.

5.1.2. Result

Figures 6 shows the performance comparison across all models for MAE. We note that ResNet-34 shows the best performance, with its MAE being only 1/5 that of the baseline model. Additionally, most SOTA models have an accuracy and R-Squared score of 1, as detailed in the appendix.

Figures 7 presents diagram comparing the ground truth versus the predicted values for both the ResNet-18 and the baseline model. It illustrates a better accuracy of the ResNet-18 model, as evidenced by the majority of its prediction points falling within the ±1 cm error margin, in contrast to the baseline model.

In essence, all models trained on our dataset, perform well. This high performance across various models and evaluation metrics underscores the high quality of our synthetic data and its applicability in real-world scenarios.

5.2. Experiment 2: Transfer Learning

In addition to validating our synthetic dataset, we conducted a transfer learning experiment to demonstrate our model's potential in real-world applications. Our goal is to employ transfer learning to make model training for height prediction in real-world settings more efficient. We anticipate that a model pre-trained on our synthetic data will transfer learned features to a real-world human height dataset, yielding better performance with fewer epochs of training under the same hyper-parameter settings.

5.2.1. Set-up

We apply transfer learning for ResNet34 in following ways:

- Trained directly on real-human data by updating all parameters (resnet34_all).
- Trained directly on real-human data by updating only the readout sequence (resnet34_readout).
- Pre-trained on ImageNet-1K and applying transfer learning on real-human data by updating all parameters (resnet34_1k_all).
- Pre-trained on ImageNet-1K and applying transfer learning on real-human data by updating only the readout sequence (resnet34_1k_readout).
- Pre-trained on FigureSynth and applying transfer learning on real-human data by updating all parameters (resnet34_syn_all).
- Pre-trained on FigureSynth and applying transfer learning on real-human data by updating only the readout sequence (resnet34_syn_readout).

For all approaches, we apply consistent and similar hyper-parameters to the previous experiment, where we validated FigureSynth. The difference is that we trained the model for more epochs—250 in total on the real human dataset. For model pre-training, instead of training on depth maps, we trained on RGB images from FigureSynth. Regarding

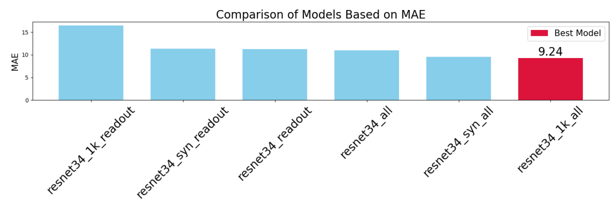

Figure 8. MAE Performance Comparison

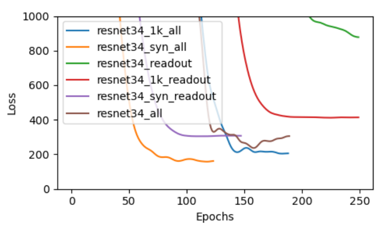

Figure 9. Validation Loss Comparison

data augmentation, we apply a horizontal flip on the fly. In addition to MAE, accuracy, and R-squared, we also employ validation loss (MSE) curve to assess training efficiency.

The real-human dataset (Bhandari, 2020) contains 380 images of humans with their height labels. We use 80% of the data for training, 10% for validation, and 10% for testing. We use zero-padding to fill images to a square format and transform each image to a size of $3 \times 224 \times 224$.

5.2.2. Results

Figures 8 show the transfer learning results for ResNet34 using mean absolute error. The diagrams indicate that ResNet34 pre-trained on FigureSynth and ImageNet-1K and fine-tuned on the real-human dataset achieves the best performance compared to other transfer learning methods.

Figure 9 shows that ResNet34 pre-trained on FigureSynth and fine-tuned either on all parameters or just the readout sequence (represented by orange and purple curves) converges faster than other transfer learning methods. The model fine-tuned on all parameters yields the best validation performance. The green curve, representing ResNet34 trained directly on the real-human dataset without pre-training, indicates that the model struggles to learn effectively.

In summary, the real-human dataset is challenging for deep neural networks to learn from scratch. Pre-training ResNet34 on FigureSynth not only substantially improves performance during transfer learning but also achieves this performance efficiently with fewer training epochs. This demonstrates that FigureSynth is a valid dataset for real-world applications with good generalization capabilities.

6. Discussion

Our experiment has shown that with the synthetic dataset, we are able to train and improve various computer vision models for predicting subject height. This shows the effectiveness and usefulness of the synthetic dataset in comparing and evaluating the performance of computer vision models.

For future work, we plan to enhance the realism of our synthetic dataset by increasing the variety of poses and environments. These improvements could help bridge the gap between synthetic and real-world scenarios, potentially leading to better generalization of the trained models. In terms of training and evaluation, we could explore more complex models that leverage the multi-view nature of our synthetic dataset. For instance, using a Mixture of Experts (MOE) model, where different experts specialize in different views and a gating mechanism combines their outputs, could lead to improved performance.

7. Conclusion

In this work, we presented a novel generative algorithm, FigureSynth, for creating synthetic images, depth maps, segmentation masks, and keypoints of human subjects. This synthetic data provides a scalable and ethical solution for training and benchmarking computer vision models to accurately predict a child's height, which is a critical component for the early detection of childhood malnutrition.

Through extensive experimentation, we demonstrated the effectiveness of our synthetic dataset in training state-of-the-art computer vision models, including CNNs and Vision Transformers, for height prediction task. The results demonstrated high accuracy and low mean absolute error, highlighting the potential of the synthetic data for such tasks.

By addressing the challenges of data scarcity and privacy concerns, our work clears the path for the ethical development of computer vision models for predicting the height of children, ultimately contributing to the early detection and prevention of malnutrition in children worldwide.

8. Reproducibility Statement

We provide the following links to enable reproducibility. To access the code and dataset for FigureSynth, please visit: FigureSynth. To access the code for the computer vision models and the modeling pipeline, please visit: Modeling.

Acknowledgements

We would like to thank our project partner Welthungerhilf for the inspiration and invaluable support.

References

POV-Ray: Persistence of Vision Raytracer. https://www.povray.org/. Accessed: 2024-05-03.

BenAbdelkader, C. and Yacoob, Y. Statistical body height estimation from a single image. In *2008 8th IEEE international conference on automatic face gesture recognition*, pp. 1–7. IEEE, 2008.

Bhandari, V. Height-weight images, 2020. URL https://www.kaggle.com/datasets/virenbr11/height-weight-images. Accessed: 2024-05-24.

Dosovitskiy, A., Ros, G., Codevilla, F., Lopez, A., and Koltun, V. Carla: An open urban driving simulator. In *Proceedings of the 1st Annual Conference on Robot Learning*, pp. 1–16, 2017.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., and Houlsby, N. An image is worth 16x16 words: Transformers for image recognition at scale. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2020. URL https://openreview.net/forum?id=YicbFdNTTy.

Ezzat, M., Albassam, E., Aldajani, E., Alaskar, R., and Devol, E. Implementation of new indicators of pediatric malnutrition and comparison to previous indicators. *International Journal of Pediatrics and Adolescent Medicine*, 9(4):216–224, 2022. doi: 10.1016/j.ijpam.2022.12.003.

Ghosh, P. G. Top ethical issues with ai and machine learning. *DATAVERSITY*, 2023.

Hansen, R. K. Ai image generator: This is someone thinking about data ethics. *Dataetisk Tænkehandletank*, 2023.

Hasegawa, J., Ito, Y. M., and Yamauchi, T. Development of a screening tool to predict malnutrition among children under two years old in zambia. *Global Health Action*, 10(1): 1339981, 2017. doi: 10.1080/16549716.2017.1339981.

Henstridge, J. Pycairo. Python package. URL https://www.cairographics.org/pycairo/. Accessed: 2024-05-04.

Kubric. Kubric: A scalable dataset generator. *arXiv preprint arXiv:1904.02463*, 2019.

Lazzerini, M. and Lassi, M. R. An overview of methods to measure child growth and body composition in the context of improving child health. *Maternal & Child Nutrition*, 15(S1):e12784, 2019. doi: 10.1111/mcn.12784.

Lee, K.-Z. A simple calibration approach to single view height estimation. In *2012 Ninth Conference on Computer and Robot Vision*, pp. 161–166, 2012.

Li, J., Sun, M., Chen, H.-C., Li, Z., and Jia, W. Anthropometric measurements from multi-view images. In *2012 38th Annual Northeast Bioengineering Conference (NEBEC)*, pp. 426–427, 2012.

Lurch, N., Hrkac, T., Gardlo, I., Hubert, Z., and Paull, L. 3d pose estimation using synthetic data over monocular depth images. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pp. 2833–2842, 2020.

Mikula, A. L., Hetzel, S. J., Binkley, N., and Anderson, P. A. Clinical height measurements are unreliable: a call for improvement. *Osteoporosis International*, 27(10): 3041–3047, 2016. doi: 10.1007/s00198-016-3635-2.

OpenAI, e. a. Learning dexterous in-hand manipulation. *ArXiv*, abs/1808.00177, 2018.

Peng, X., Bai, Q., Xia, X., Huang, Z., Saenko, K., and Chang, B. Synthetic data for deep learning. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 0–0, 2018.

Rajappan, D. R. E., Sanith, S., and Subbulakshmi, S. Malnutrition detection using deep learning models. In *2023 4th IEEE Global Conference for Advancement in Technology (GCAT)*, pp. 1–8. IEEE, 2023.

Ramirez, P. A., Denman, S., Fookes, C., and Sridharan, S. Deep learning for sensor fusion: Review, challenges and research directions. *Sensors*, 21(15):5032, 2021. doi: 10.3390/s21155032. URL https://doi.org/10.3390/s21155032.

Rhodin, H., Salzmann, M., and Fua, P. Unsupervised geometry-aware representation learning for 3d human pose estimation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 765–782, 2018.

Shu, H., Ren, L., Pan, L., Huang, D., Lu, H., and Wang, W. Single image based infant body height and weight estimation. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pp. 6052–6059, 2023. doi: 10.1109/CVPRW59228.2023.00644.

Su, H., Maji, S., Kalogerakis, E., and Learned-Miller, E. Multi-view convolutional neural networks for 3d shape recognition. In *Proceedings of the IEEE international conference on computer vision*, pp. 945–953, 2015.

Textures, O. G. 100 seamless textures. OpenGameArt.org, 2024. URL https://opengameart.org/content/100-seamless-textures. Public Domain. Available at: https://opengameart.org/content/100-seamless-textures.

Trivedi, A., Jain, M., Gupta, N. K., Hinsche, M., Singh, P., Matiaschek, M., Behrens, T., et al. Height estimation of children under five years using depth images. In *2021 43rd Annual International Conference of the IEEE Engineering in Medicine Biology Society (EMBC)*, pp. 3886–3889. IEEE, 2021.

Varol, G., Romero, J., Martin, X., Mahmood, N., Black, M. J., Laptev, I., and Schmid, C. Learning from synthetic humans. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 109–117, 2017.

Yin, F. and Zhou, S. Accurate estimation of body height from a single depth image via a four-stage developing network. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 8264–8273, 2020.

A. Details for Model Training Experiments

Height Prediction with CNN Backbone The output of pretrained Convolutional Neural Network models is processed through a custom fully connected network. For all models, we have standardized the fully connected network architecture to follow the structure of 512, 256, and finally 1 neuron. This means that the last linear layer outputs a scalar value, which represents the predicted height. Additionally, this same fully connected network setting is also applied to our baseline 3-layer CNN model.

Height Prediction with ViT Backbone The Vision Transformer (ViT) processes the input image into an array of feature vectors corresponding to the number of image patches, plus an additional class token `[CLS]`. In alignment with the fully connected network used for CNN backbone models, the output head of the ViT applies the same architecture to the `[CLS]` token. This results in a final output that is a single scalar value, representing the predicted height.

Training Procedure We trained on Google Colab with an `A100` GPU. All models, except for the baseline CNN, are pre-trained, and we performed transfer learning by fine-tuning all parameters. Each model was trained for 100 epochs with a learning rate of 0.00002 using the Adam optimizer and a batch size of 128. The read-out fully connected layer has a dropout rate of 0.1. Early stopping with a patience of 30 epochs is implemented, which is triggered when the model's validation loss does not improve. The best model is saved based on the lowest validation mean squared error (MSE) loss.

In addition to the training parameters, we applied data augmentation techniques on-the-fly exclusively for the training set. These techniques include random rotations within a range of ±15 degrees and random horizontal flips to introduce variability and prevent over-fitting. Furthermore, we normalized the data with a mean of [0.3568, 0.3568, 0.3568] and a standard deviation of [0.3512, 0.3512, 0.3512]. This normalization process adjusts the pixel values of the images so that the dataset has a mean of zero and a standard deviation of one, which helps in accelerating the training process and achieving more consistent performance.

Evaluation Metrics We report and compare model performance with 3 different metrics: Mean-Absolute Error, One-centimeter Tolerance Accuracy and R-squared.

- Mean-Absolute Error: $\frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}_i|$

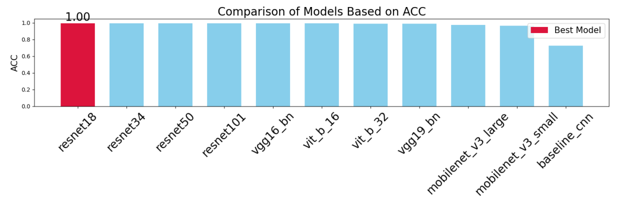

Figure 10. Accuracy Performance Comparison

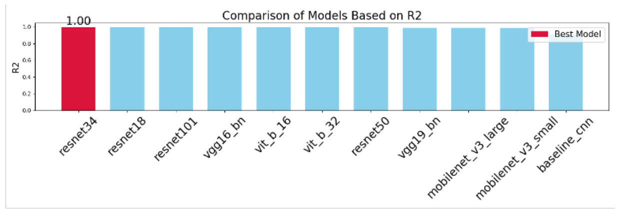

Figure 11. R2 Performance Comparison

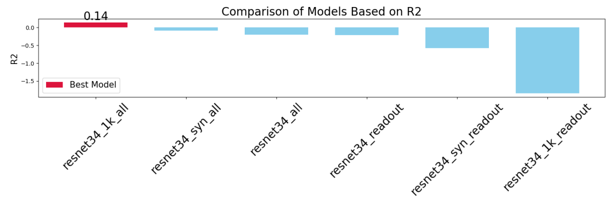

Figure 12. R Squared Performance Comparison

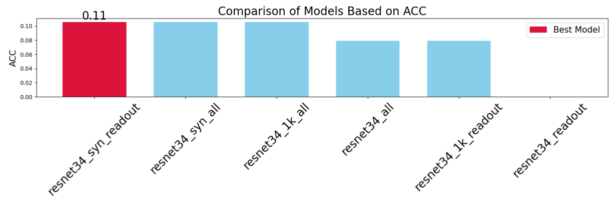

Figure 13. Accuracy Performance Comparison

- One-centimeter Tolerance Accuracy:

$$\frac{1}{N}\sum_{i=1}^{N}\mathbb{1}(|y_i-\hat{y}_i|<1\text{cm})$$

- R-squared:

$$1-\frac{\sum_{i=1}^{N}(y_i-\hat{y}_i)^2}{\sum_{i=1}^{N}(y_i-\bar{y})^2}$$

Validating Experiment results Figures 10 and 11 show the performance comparison across all models for Accuracy and R-squared respectively. We note that almost all models except the baseline have exceptional accuracy and R-squared of almost 1.0. Specifically, ResNet-18 having the highest accuracy and ResNet-34 having the highest R-squared.

Transfer Learning results For the transfer learning experiment where we train ResNet34 using different approaches, the R-Squared and accuracy metrics show similar trends, as illustrated in Figures 12 and 13

B. Ablation Studies

Mutation of ViT We further experimented with a slight modification of the Vision Transformer architecture by jointly predicting on multiple views simulated from different angles of the same subject. We implemented two versions of this collective decision algorithm:

- **Simple Aggregation:** An aggregator takes in output features from all views of a single subject, combines the feature representations by averaging across simulated views, and applies the MLP regression head to the combined feature. Algorithm workflow is demonstrated as Figure 14.
- **Aggregation with Concatenation:** An aggregator first combines information from multiple views, and then concatenates the aggregated feature with individual features from each view. The MLP regression head acts on a combination of combined and individual features. Algorithm workflow is demonstrated as Figure 15.

We trained and validated the two algorithms with inputs of raw simulated subject photos in RGB format, and directly predicting height values. We realized a performance increase from 46% accuracy on non-mutated, vanilla ViT to 55% accuracy with *Simple Aggregation* and 65% accuracy with *Aggregation with Concatenation*.

Performance boost from vanilla Vision Transformer to *Simple Aggregation* demonstrates the effectiveness of information sharing among different simulated viewpoints, and a further increase with the concatenation operation implies the value of larger feature dimensions and fusion of individual and collective information.

However, we would also like to note the difference between vanilla Vision Transformer performance reported here and

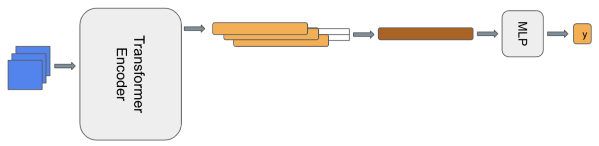

Figure 14. Simple Aggregator Pipeline

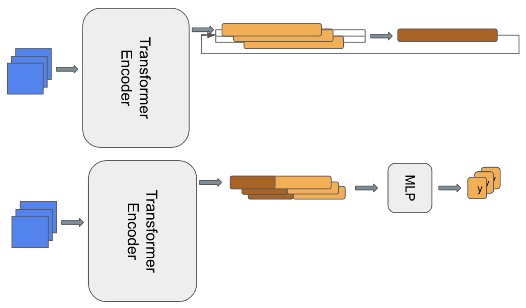

Figure 15. Aggregation with Concatenation

that shown in Figure 10. The latter one was trained on depth images, while the former one was trained with raw photos. A performance degradation with `RGB` input may be due to depth maps having less noisy inputs. One could regard the depth map as the segmentation map of the subject from raw `RGB` images. The use of depth maps avoids additional overhead for the model backbone to re-segment the subject, which would lead to higher prediction accuracy, especially when the pre-loaded ViT backbone weights were initially trained for classification tasks.

C. Metadata Files

FigureSynth provides a metadata file associated with each rendered RGB image, depth map, and segmentation map. These files contain precise values for all of the scene's features, including the child's height, in simple text format.

```
child height (cm): 138.55981802462807
seed: 1
head_size: 2.5488135039273248
skin_color: [1.          0.95529137  0.37316718]
shorts_color: [1.          0.07863797  1.          ]
mat_texture: 171
camera_distance: 6.188992975388076
rotation_adj: -0.429241808987133
right_arm_angle_xy: 13.654671795124212
left_arm_angle_xy: -0.9769572171942498
right_arm_angle_z: -12.129772850064 69
left_arm_angle_z: -2.26527900963526
left_leg_angle_xy: -4.707710044024339
right_leg_angle_xy: 9.730159968843793
right_leg_angle_z: -6.390745588515077
left_leg_angle_z: 7.186853402106025
body_height: 2.4675672693585855
light_x: 3.0091075197964425
light_y: 0.2047747955120478
light1: 0.7752156710832722
light2: 0.8044428583081418
light3: 0.6656158336600857
light4: 0.6761612606143075
backcolor1: 0.7586156243223572
backcolor2: 0.10590760718779213
backcolor3: 0.4736004193466574
haircolor_red: 1.0
haircolor_green: 0.6089630164928302
haircolor_blue: 0.559124280628426
eye_color_index: 2
smile_factor: -21.06945572323671
mat_rotate_y: -0.7034359688271437
mat_rotate_z: -2.101950796897901
child_rotate_y: -4.696653025600872
child_rotate_z: -3.245497790353857
child_rotate_x: -0.013232273685800534
hair_count: 3634.7442832352676
hair_length: 0.1359959991928982
image_noise_level: 4.904507663073059
child_shift_x: 0.040234858317398425
child_shift_y: -0.04007196496410262
child_shift_z: 0.04698090677467488
head_height: 4.379177397304079
child_height: 6.927990901231404


keypoint_head_top_3d: (0, 6.927990901231404, 0)
keypoint_head_top_2d: [ 0.5         -0.06258539]
keypoint_right_hand_3d: [1.46428544  1.67462614  0.          ]
keypoint_right_hand_2d: [0.59401776  0.39838777]
keypoint_left_hand_3d: [-1.48621079   2.20292239  -0.4         ]
keypoint_left_hand_2d: [0.40503942  0.43565807]
keypoint_right_foot_3d: (0.3674759165589824,
      -0.15772090320115337, 0.0)
keypoint_right_foot_2d: [0.52157343  0.68697878]
keypoint_left_foot_3d: (-0.45565292322531353,
      -0.14781180773030586, 0.0)
keypoint_left_foot_2d: [0.47618232  0.68637685]
```

Listing 1. Example portion of a metadata file associated with a single view of a rendered scene