Case 2 Report: 3D Part Similarity with CNNs

Designing Smart and Intelligent Products

Team 4: Nitya Mathur, Yingxin Zhang, Giovanni Natasha, Krishna Kalla
Date: October 12. 2024

Executive Summary

Engineering simulations are becoming essential tools across various industries. These simulations support engineers and designers in visualizing, testing, and refining designs before they reach production. It reduces development time and costs while ensuring product quality. However, traditional design workflows can be time-consuming, and current tools often lack the intuitive capabilities that design teams require to innovate rapidly. Integrating artificial intelligence (AI) and machine learning (ML) into these workflows has significant potential to transform the design process, making it more efficient, intuitive, and powerful.

We aim to leverage AI/ML to empower organizations to design more intuitively and efficiently, fostering innovation and competitive advantage in product development. A key opportunity that AI can provide is democratizing engineering simulations, making them accessible and intuitive for more members of product development teams. As simulations become increasingly integral for design, we recognize the need for tools that enhance usability and accelerate workflows. By integrating AI and ML into the design process, we aim to transform simulations from complex, specialist tools into broadly usable resources that empower teams to innovate more freely.

In pursuit of these goals, we developed a 3D Convolutional Neural Network (CNN) pair similarity model, achieving a high 97% accuracy in classifying and detecting 3D objects. This accuracy aligns with industry standards for high-performing design tools, positioning our model as a valuable asset in enhancing digital prototyping and simulation accuracy. The model not only supports traditional design tasks but also opens new avenues for intuitive decision-making for teams throughout the product life cycle.

Business Considerations

The engineering simulation market is projected to reach $33.5 billion by 2028, with a 13.5% CAGR (Markets and Markets, 2023). Engineering simulations are not only integral to the product design workflow, but offer organizations a smart return on investment by speeding up the development time by 9 times, and increasing profit margin by 15% (The Aberdeen Group). However, the main barrier to entry for the simulation market is accessibility of using the software tools. According to industry experts, simulation providers will no longer be able to capture the market by focusing on reducing the accuracy and speed of their software (Siemens). They will also need to lower the complexity of using their simulations, thus highlighting the need to democratize simulations.

Our value proposition for the 3D CNN model is that it will cater specifically to engineering simulation providers, enhancing the user experience of their clients in various impactful ways. According to a survey by McKinsey of 176 respondents, encompassing both simulation providers and users, the top value driver organizations wanted from their simulations that emerged was enabling a faster time-to-market for their products (Ragani & Stein, 2023). This finding denotes an opportunity to leverage AI to accelerate product development workflows to support this value driver. By integrating advanced AI-driven object recognition, our model transforms simulations into a more accessible resource, reducing technical barriers and empowering wider use across product development teams. The following key considerations highlight the primary business drivers of this solution:

Use Cases

The 3D object recognition software identifies individual parts within prototypes, enhancing accuracy in design simulations. This capability enables teams to detect and address design issues early, improving product reliability and performance.

Value Drivers

1) Intuitive User Experience
   By lowering the learning curve and creating a more interactive simulation experience, the model expands the potential user base and allows non-specialists to engage meaningfully with design processes. For example, business specialists could better understand all the specific parts required for the design assembly which could help in financial decision-making and supply chain assessments.

2) Accelerating Design Workflow
   The model's accessible interface and automation capabilities help reduce barriers to simulation, enabling teams to prototype, test, and refine designs faster. Designers who lack the knowledge of highly-trained engineers with simulations could be able to iterate on designs with part recognition support from AI-powered object detection. This accelerated workflow shortens the time-to-market, helping organizations respond quickly to market demands.

3) Enabling Cross-team Collaboration
   The democratized approach to simulation encourages collaboration across disciplines, allowing team members from different backgrounds to contribute insights and ideas while interacting with the simulation designs. This inclusivity not only drives creative solutions but also strengthens decision-making across the product lifecycle.

AI/ML Implementation

Hypothesis

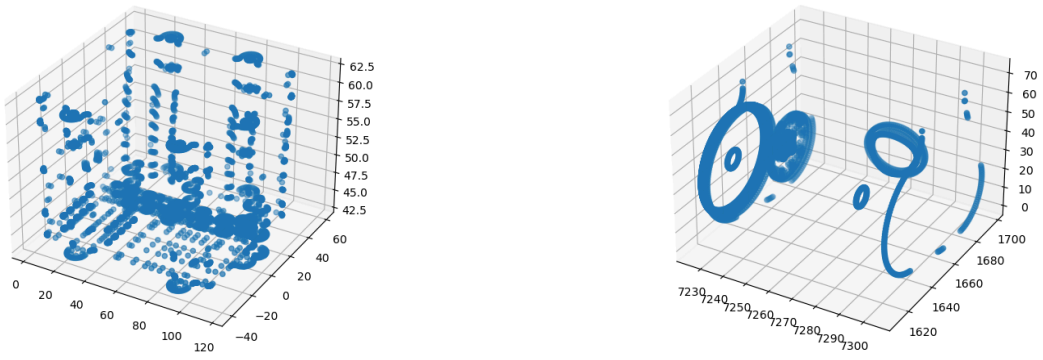
The goal of our project was to determine if we could build a Convolutional Neural Network (CNN) model capable of achieving over 95% accuracy in detecting similarities between 3D objects for industrial utilization. As the use case of this model was designed for industrial designers, 95% accuracy is enough for this case. The hypothesis was tested by developing a model architecture that could handle complex 3D inputs, while also being efficient enough to perform well in real-world scenarios. We refined the model through iterative parameter tuning and optimization, with a target of balancing performance, resource consumption, and scalability.

Data Analysis

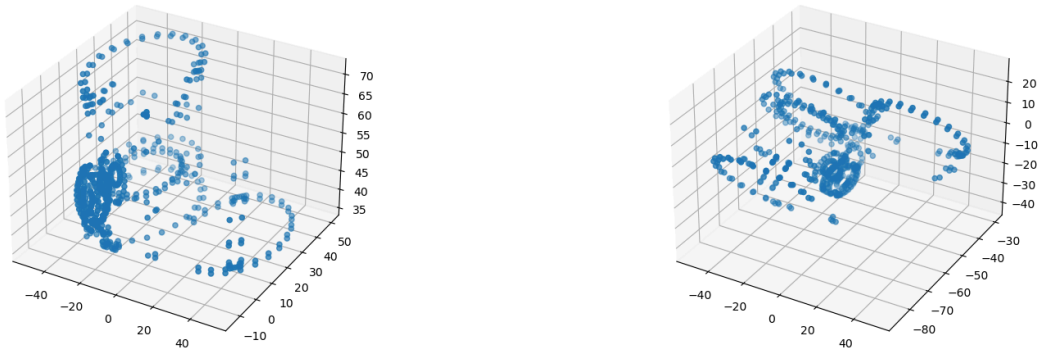
We were provided with 31,867 data points, where each point consisted of:

1. Image 1
2. Image 2
3. Label: 0 meant that the two images were dissimilar while 1 implied they were similar. The images could be of the same items but with changes such as from different perspectives or rotated - in that case the labels are still 1.

Some examples are shown below.



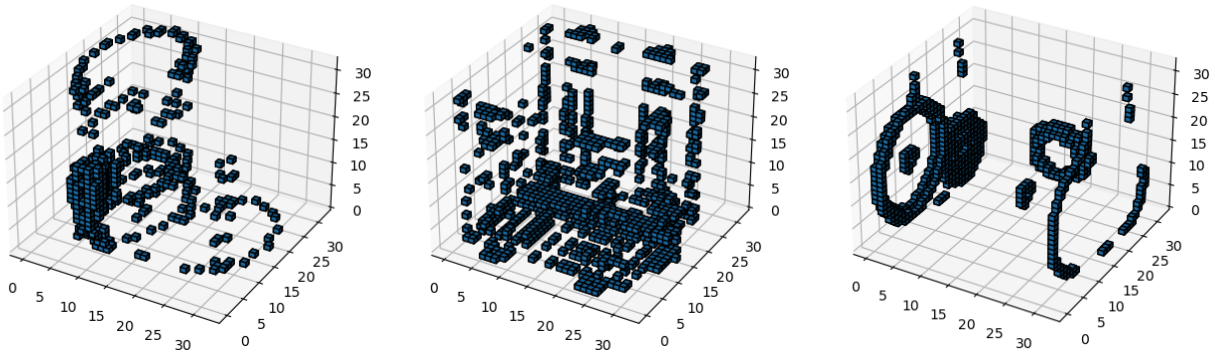
Label 0: Dissimilar (File 4000.npy)



Label 0: Similar (File 8000.npy)

The dataset was evenly distributed between similar and dissimilar images. This ensures that there is no bias while training.

To create an image into an appropriate format for the neural network, we performed voxelization. We converted each point cloud representation to a **32x32x32 resolution voxel grid.** We picked 32 as a dimension since 64 was too heavy to process on our limited computational power. Increasing the voxel size to 64 may provide better results for the future. Voxelization divides the 3D space into discrete cubes or voxels, allowing the model to process volumetric data effectively.



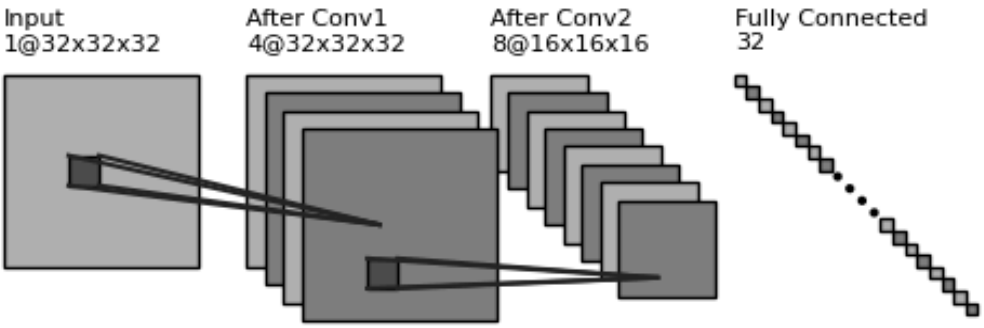
Voxelized Images

We split the dataset into 70% training, 15% validation and 15% test to evaluate model performance effectively. We also used a DataLoader to implement batching and pass a subset of data through the model at a time. This is useful when parameter tuning because passing all of the data through the model takes a long time.

Model Architecture

We used a **Siamese encoder architecture** for our CNN. In a Siamese network, images are passed through 2 subnets that share their weights and architectures. Then we take the difference of the 2 models and run in through a sigmoid activation function to get the similarity score. We picked this architecture since it is commonly used for similarity-based tasks. They are advantageous since they generalize well when presented with new classes during inference, which makes our model flexible if presented with new data later in the product life cycle.

Considering the vast amount of data available to train on, we first implemented a base model. The base model summary is as follows:



Base Model Architecture

1. Input Data

The input data for the model consists of 3D voxel grids with dimensions 1×32×32×321, where 1 represents the number of input channels (a single 3D voxelized representation).

2. **Encoder3D Architecture**

The Encoder3D subnetwork is the core of the base model. Both inputs are passed through the same shared network. The encoder consists of the following layers:

- **First 3D Convolutional Layer**

The first convolutional layer learns 4 feature maps from the input data, with each feature map representing different spatial patterns or structures found in the 3D voxel grid. The kernel scans the 3D input with a 3x3x3 filter and applies ReLU activation to introduce non-linearity, allowing the model to capture complex relationships in the data. Padding is 1 to maintain the spatial resolution of the voxel grid

- **Second 3D Convolutional Layer**

The second convolutional layer increases the number of feature maps to 8, further refining the learned features from the first layer. This layer reduces the spatial dimensions of the input due to a stride of 2, which effectively halves the grid size, making it 16x16x16 while learning deeper representations of the voxelized input.

- Flattening Layer

After the second convolutional layer, the output is flattened to convert the 3D feature maps into a 1D vector for the fully connected layer. The size of the flattened feature vector is 8×16×16×16=32,7688.

- Fully Connected Layer

The flattened feature vector is passed through a fully connected layer that reduces the feature vector size to 32 dimensions. This compact representation captures the essential features needed for comparing two voxelized inputs.

3. Siamese Network

After both inputs are processed by the Encoder3D subnetworks, the resulting feature vectors from the two inputs are compared. The comparison is done by calculating the absolute difference between the two feature vectors. This difference represents how similar or different the two inputs are based on their learned features.

4. Output Layer

The combined feature vector is passed through a fully connected output linear layer of 32 dimensions. A sigmoid activation function is applied to this output.

5. Loss Function

The model is trained using **binary cross-entropy loss** (BCELoss). The goal is to minimize the difference between the predicted similarity score and the true label.

6. Training Process

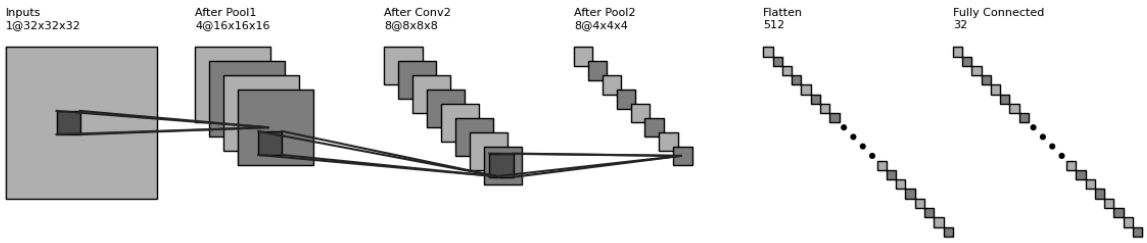
The model is trained with a **batch size of 512**. During training, the parameters of the convolutional layers and fully connected layers are adjusted to minimize the binary cross-entropy loss, allowing the model to learn how to differentiate between similar and dissimilar pairs of voxelized inputs.

Another representation that highlights the parameters is as follows:

```
==========================================================================
Layer (type:depth-idx)                   Param #
==========================================================================
├─Encoder3D: 1-1                         --
|    └─Conv3d: 2-1                       224
|    └─Conv3d: 2-2                       3,472
|    └─Linear: 2-3                      2,097,184
├─Linear: 1-2                            33
==========================================================================
Total params: 2,100,913
Trainable params: 2,100,913
Non-trainable params: 0
==========================================================================
==========================================================================
Layer (type:depth-idx)                   Param #
==========================================================================
├─Encoder3D: 1-1                         --
|    └─Conv3d: 2-1                       224
|    └─Conv3d: 2-2                       3,472
|    └─Linear: 2-3                      2,097,184
├─Linear: 1-2                            33
==========================================================================
Total params: 2,100,913
Trainable params: 2,100,913
Non-trainable params: 0
==========================================================================
```

We also trained **more complex models** by gradually adding on features to optimize the training and validation loss curves as well as the final testing accuracy.

An example for a more complex architecture we implemented is as follows:



This model has layers for dropout, pooling, and more techniques to prevent overfitting. This complex model has more layers that progressively reduce the spatial dimensions, making it both more powerful and more efficient in extracting features at different levels of detail.

Parameter Tuning

To achieve optimal performance, we adjusted following key parameters, ensuring a balance between model efficiency and accuracy.

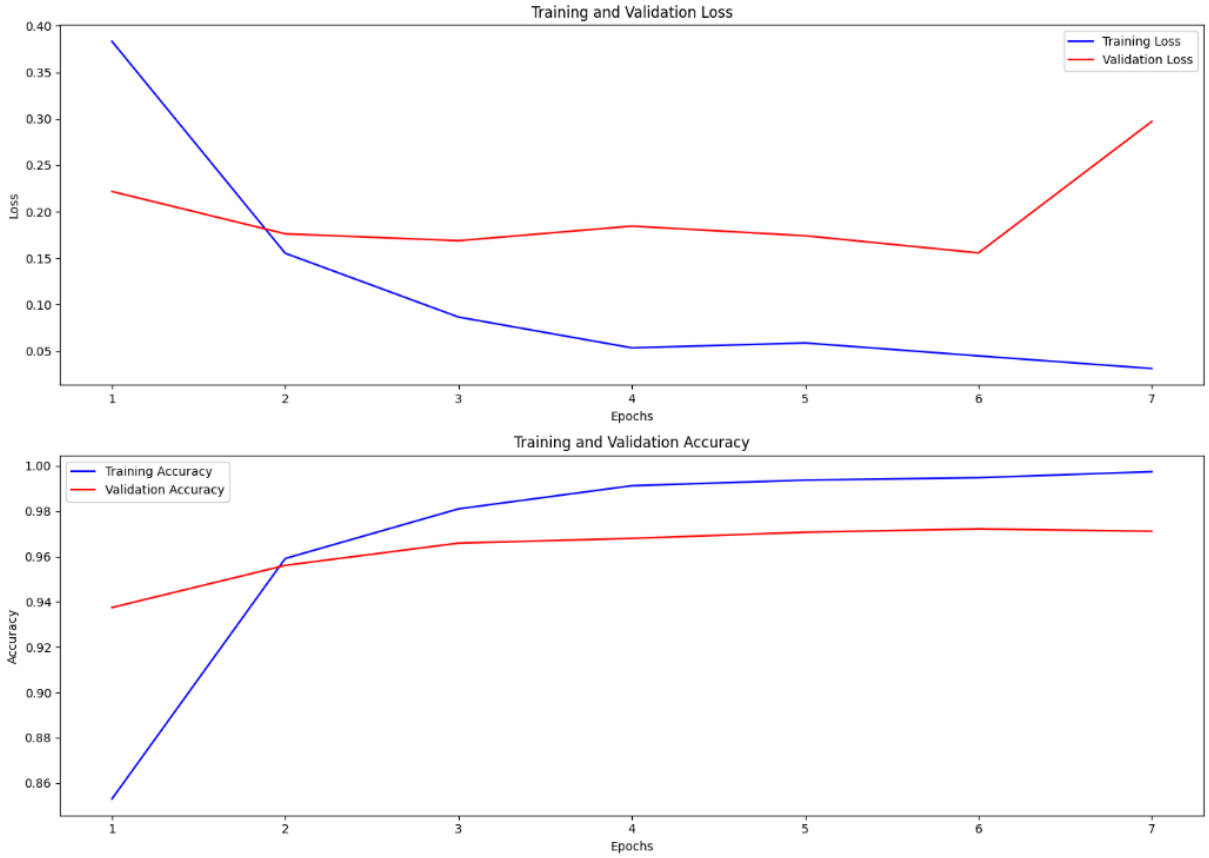
- **Learning Rate**: The learning rate significantly impacts the speed and stability of convergence. After experimentation with values such as 0.01, 0.001, 0.002, and 0.0005, we found that a learning rate of 0.001 provided the best balance between stable and efficient convergence. Higher values (like 0.01) led to instability, while lower values (below 0.001) slowed down the training process considerably, making them inefficient for our large dataset. Unlike them, the chosen rate, 0.001, successfully ensured that we maintained training momentum without sacrificing stability.
- **Batch Size**: Given the size of our dataset (over 30,000 files), batch size was a critical factor in balancing computational efficiency and model generalization. We tested batch sizes of 8, 128, and 512, and found that a batch size of 512 provided the best trade-off. It allowed for faster computation while maintaining a high level of accuracy, ensuring that the model could scale effectively in real-world applications without compromising performance.
- **Number of Layers**: The depth of the network plays a crucial role in feature extraction. We explored 2, 3, and 4-layer architectures. While deeper networks (3-4 layers) enhanced feature extraction, we observed a slight decrease in prediction accuracy with results ranging from 92% to 95%, and longer training time. After evaluating the performance impact and considering business objectives, we opted for a 2-layer architecture, which satisfies both the accuracy and computational efficiency requirements.
- **Preventing Overfitting**: Initially, we encountered overfitting after the first few epochs—while training accuracy improved, testing accuracy plateaued. To address this, we applied several techniques, including resetting model weights, weight decay (set to 0.00001), and dropout (set at 0.3 or 0.4). Although dropout is often an effective technique, in this case, it slowed training significantly without yielding any meaningful improvement in accuracy or overfitting. Therefore, our final model employed **reset weight** (set to 0.00001) and **weight decay**, successfully mitigating overfitting without compromising training speed.

These tuning adjustments lead to a model that achieved **> 97% testing accuracy**, surpassing our initial hypothesis of 95%.

Model Evaluation

We implemented a maximum of 15 epochs, with a mechanism to automatically save the best-performing model for each epoch based on the validation accuracy for each epoch that halts the training process when further improvements in the validation results are no longer observed. This dynamic approach minimizes manual adjustment of the number of epochs, while allowing the model sufficient time to learn, reducing the risk of overfitting and unnecessary computation.

The best model achieved an accuracy of 97.47%. We plotted the following curves to assess the model:



Training and Validation Loss vs Epochs;
Training and Validation Accuracy vs Epochs

Analyzing the above graph, If we focus on the training and validation loss graphs at the top, we can observe that both models show a decrease in loss over time, which indicates the model is learning well. Clear signs of overfitting are not observed, (highly) probably because the model gets saved before the curves diverge.

Computing Resources

Efficient resource management was critical to ensure the optimal performance of our model without overburdening the hardware or increasing computing or infrastructure costs. We evaluated both our current hardware setup and advanced GPUs to confirm the feasibility of deploying our model in real-world scenarios.

We performed training and model tuning on an M2 Pro chip with 32GB of memory and a 512GB SSD. Depending on key parameters—such as the number of epochs, layers, batch size, and techniques to prevent overfitting—training times ranged between 5 to 15 minutes per round. This demonstrated that even with a standard hardware setup, our model performed efficiently. As many users may have access to more advanced hardware, training and running our model within their environments would not only be feasible but also seamless to implement.

For optimal performance, advanced GPUs can be leveraged to achieve state-of-the-art results. High-end GPUs, like the Nvidia A100 or RTX 3090, are capable of handling the more demanding phases of model training. These GPUs can execute multiple teraflops (TFLOPS) per second, which significantly accelerates the training process and allows the management of larger datasets. When using these GPUs, our model processes around 79,429,632 FLOPs during training and around 26,476,544 FLOPs during inference. In terms of memory usage, each batch of 512 samples requires approximately 900MB during training and 500MB during inference. With the power of these advanced GPUs, our model can efficiently scale while maintaining high speed and computational performance. This ensures that whether running on a standard setup or utilizing advanced hardware, our model remains adaptable, scalable, and cost-effective for real-world applications.

Both the test accuracy and computer resources evaluation outcomes demonstrate our model's capability to effectively address the designer use case, providing both high accuracy and efficient performance suitable for real-world application scenarios.

Discussion

To conclude, a simpler model with less layers performs better than a more complex model. Why is this? Possible reasons are:

- The complex model, despite being deeper and having more layers, might have introduced additional complexity that wasn't necessary for this particular dataset, leading to reduced test performance. A fix would be more quantity and variance of data.
- We could not hyperparameter tune the number of feature maps that the layers are producing. With the first layer capturing only 4 sets of features, important information could have been lost.

The team learnt immensely about the realistic process of building a CNN. What surprised us was the number of variables that have to be juggled with to produce an optimum result, which would be exponentially more difficult with a multi-label classification problem.

Business Trade-Offs

When developing a model for practical, real-world applications like assisting designers, understanding the business trade-offs becomes just as important as technical optimization. As we fine-tuned our model for 3D object detection, we identified four key business trade-offs that link model training and tuning with future product development.

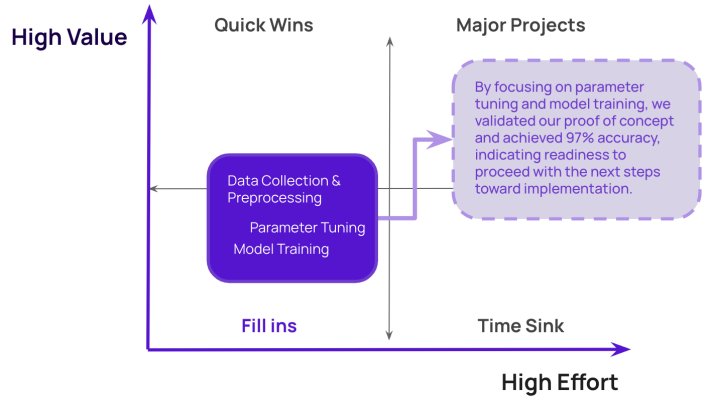
- **Accuracy vs. Cost:**
  Higher accuracy often comes at the cost of greater computational power. While a model with over 95% accuracy significantly improves the designer's workflow by identifying 3D object similarities, maintaining this precision requires more advanced hardware, such as high-end GPUs, leading to increased training costs. For designers working on rapid product iterations, balancing between high accuracy and acceptable infrastructure costs becomes a crucial decision in scaling the model.
- **Real-Time Processing vs. Resource Consumption:**
  For designers, the ability to process 3D objects in real-time is critical for quick iteration and fast feedback. However, real-time processing demands significant computational resources, particularly during model inference. This means using more memory and GPU/CPU power to achieve the rapid responses that designers need. While this enhances productivity, the trade-off between the added expense of high-end resources and the value of faster iterations will be assessed when customers make purchasing decisions.
- **Data Requirements vs. Scalability:**
  Scaling the model to adapt across different design iterations or product prototypes requires large, diverse datasets. Designers often work with varied materials, shapes, and object types, which necessitates a model capable of learning from a broad range of data. As the model scales, so do the data requirements, increasing the need for labeled datasets. While this ensures the model's adaptability, it also adds complexity to data collection and preparation, affecting the overall scalability of the solution.
- **Flexibility vs. Specialization:**
  For designers, flexibility is key. The model must adapt to multiple product designs, materials, and evolving customer demands, ensuring it can handle different projects over time. However, specialization often leads to better performance for specific tasks. A highly specialized model could yield higher accuracy for a particular product but may require significant retraining for new designs. Therefore, the trade-off here is between building a flexible, versatile model that serves a broader range of use cases versus a specialized model that excels in a narrower scope but requires more maintenance.

Limitations

While our model performed well in terms of accuracy, there are some limitations to its use cases based on the current implementation. First, the model excels at identifying similarities between 3D objects but lacks the ability to label them. For tasks requiring object identification, the model would need to be restructured and retrained with labeled data. Also, to achieve effective labeling, it requires large datasets with labeled 3D items to train and fine-tuning the new model. Furthermore, while the model performs well for known 3D objects, its flexibility when applied to entirely different types of industrial accessories is still uncertain. More data and testing are needed to ensure the model can generalize to a wider variety of industrial objects.

Product Considerations

Product Prioritization Framework



For this proof of concept, we categorized it as a medium value, low effort project on a value complexity matrix. Before we can consider building a full-scale product, we had to build a proof of concept to validate our hypothesis. The preparatory scope of our project encompasses data collection and preprocessing, parameter tuning, and training the model. As a result, we have built a simple model that can achieve 97% accuracy for 3D image classification. Our achievement indicates that we are ready to proceed with product implementation and further steps to ensure the robustness of our algorithm.

Algorithm Next Steps

The CNN model currently achieves an accuracy of 97%, surpassing the desired threshold of 95%. However, there are concerns regarding potential overfitting to the validation set. To mitigate overfitting and enhance the model's generalization ability, we plan to experiment with the following techniques:

1. **Weight Decay**:
   Weight decay, also known as L2 regularization, penalizes large weights by adding a term proportional to the square of the weights to the loss function. This discourages the model from relying too heavily on any single feature and forces it to learn simpler, more robust patterns. By applying weight decay, we aimed to reduce the likelihood of overfitting and would continue experimenting with more with it to reduce overfitting.

2. **Dropout**:
   Dropout is a regularization technique where a proportion of neurons are randomly set to zero during each forward pass of training. This prevents the model from becoming overly dependent on specific neurons or features and encourages it to learn more diverse representations. By experimenting with different dropout rates, we could reduce overfitting by making the model less likely to memorize the training data and more resilient to variations in unseen data and would continue experimenting more with it to

reduce overfitting even further.

3. **Max Pooling**:
   Max pooling reduces the spatial dimensions of feature maps by selecting the maximum value from each sub-region of the input. This downsampling operation not only reduces the number of parameters in the model but also focuses on the most salient features, thereby preventing the model from learning unnecessary details. By applying max pooling, we aim to improve generalization by reducing the risk of overfitting while retaining the most important information in the data.

4. **Feature Maps (Number of Filters)**:
   Feature maps, created by convolutional layers, represent different features detected by the model at various levels of abstraction. Adjusting the number of filters in each layer can affect the model's capacity to learn. Increasing the number of feature maps allows the model to capture more detailed patterns, while reducing the number helps avoid overfitting by limiting the model's complexity. We will experiment with different numbers of feature maps to balance the model's learning capacity and its ability to generalize well on unseen data.

5. **Data Augmentation**:
   Data augmentation involves applying transformations to the training data, such as rotations, flips, zooms, and shifts, to artificially increase the diversity of the dataset. This helps the model generalize better by preventing it from overfitting to specific patterns in the training set. By introducing more variability in the training data, we can force the model to learn more robust features, ultimately improving its performance on the validation and test sets.

Business Next Steps

As part of the business rollout strategy, we will implement a **controlled rollout** approach, adhering to agile principles. This involves continuously gathering feedback, iterating on the product, and incrementally expanding the rollout. Each phase will be scaled based on an evaluation of risks, impacts, and the required effort to ensure a smooth and effective deployment.

- **Phase 1: Beta Rollout to Small, Low-Risk Clients**
  The initial rollout will target a select group of low-risk clients in smaller industries. These industries present minimal risk and offer a controlled environment to address any potential issues. They are typically more open to sharing data in exchange for personalized solutions or enhanced support, making them ideal for the beta phase. Feedback and insights from this group will be collected to identify emerging use cases and refine the product accordingly.

- **Phase 2: Expansion to Small-Scale Businesses**
  Following successful beta testing, the next phase involves a soft launch into select small-scale businesses. This phase will focus on testing the product's scalability and identifying new challenges. Feedback from this group will help in further optimizing the product while maintaining continuous learning and refinement as the rollout progresses to larger businesses.

- **Phase 3: Rollout to Medium-Scale Businesses**
  Once iterations based on small-scale feedback are complete, the rollout will expand to medium-scale businesses. This phase is crucial for testing the product's robustness and adaptability in larger environments. It ensures that the product can handle increased complexity and prepares it for wider-scale adoption.

- **Phase 4: Full Rollout to Large-Scale Enterprises**
  After ensuring successful deployment in small and medium businesses, the product will be rolled out to large-scale enterprises. At this stage, the focus will be on ensuring consistent performance, addressing long-term operational issues, and reinforcing robust support systems. Additionally, insights gathered throughout the rollout will be used to refine the business model, ensuring it aligns with market demands and is scalable for widespread adoption.

Conclusion

In conclusion, our AI-powered object detection software for simulations empowers organizations to democratize and streamline product development workflows, transforming simulations into accessible and intuitive tools for diverse product team members. By implementing a 3D CNN model with a high accuracy rate, we address the industry's need for advanced, user-friendly simulation capabilities that transcend traditional limitations. This model not only accelerates time-to-market by enhancing usability and accuracy but also fosters innovation by enabling non-specialists to engage meaningfully in the design process. Our commitment to usability and efficiency ensures a versatile solution that adapts to diverse product development needs, ultimately driving competitive advantage for organizations.

References

Aberdeen Group. (n.d.-a). The impact of strategic simulation on product profitability. https://www.cadfem.net/fileadmin/user_upload/05-cadfem-informs/resource-library/wp-impact-of-strategic-simulation-on-product-profitability.pdf

Jean-Claude Ercolanelli senior vice president, Simulation and Test Solutions. Siemens Digital Industries Software. (n.d.). https://www.sw.siemens.com/en-US/leadership/technology-leadership/jean-claude-ercolanelli/

Ragani, A. F., & Stein, J. P. (2023, June 21). Unveiling the next frontier of engineering simulation. McKinsey & Company. https://www.mckinsey.com/capabilities/operations/our-insights/unveiling-the-next-frontier-of-engineering-simulation

Simulation software market size & trends, growth analysis, industry forecast [2030]. MarketsandMarkets. (n.d.). https://www.marketsandmarkets.com/Market-Reports/simulation-software-market-263646018.html