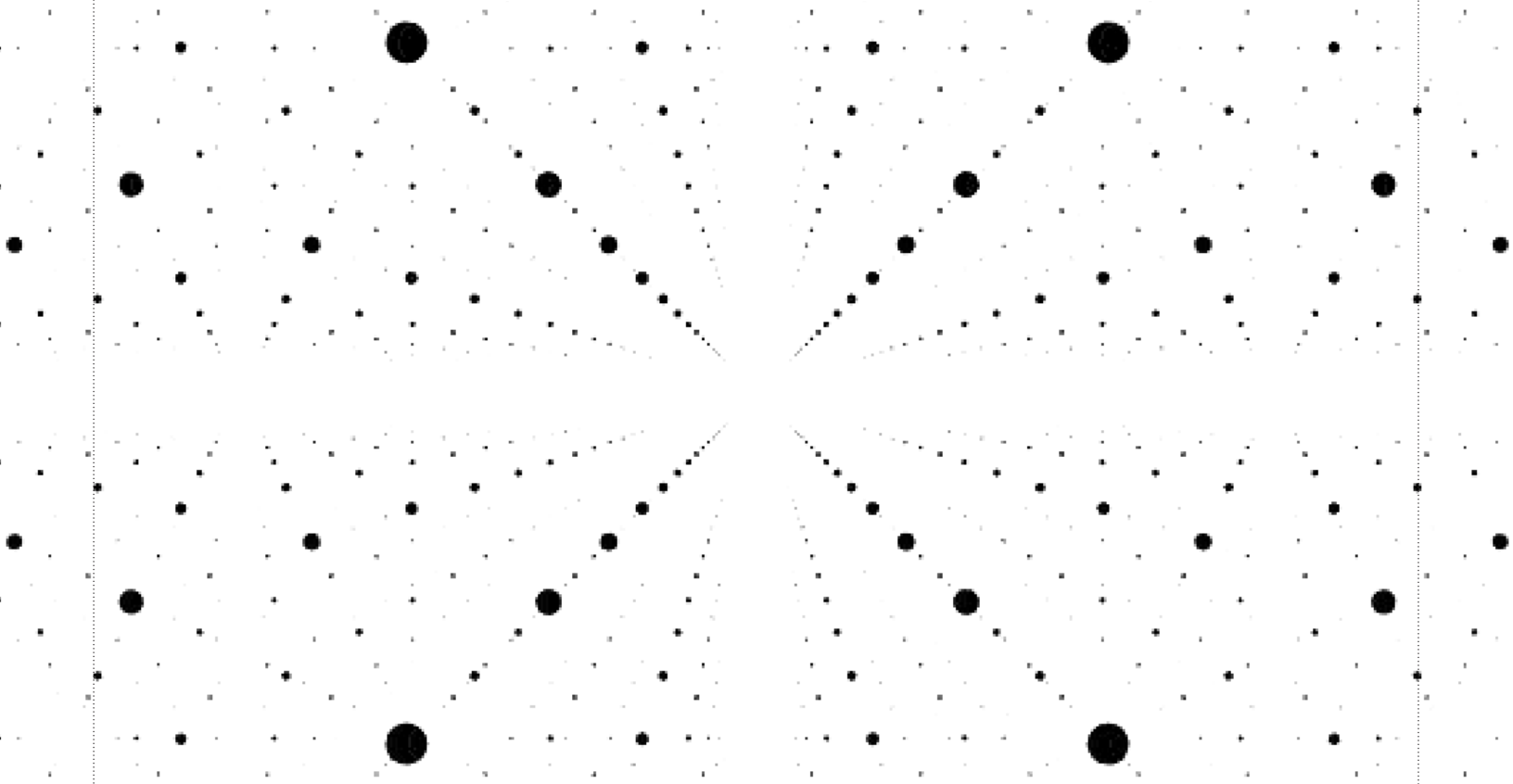
Suggestive Computer-Aided Design

Assisting Design Through Machine Learning

Stanislas Chaillou | Harvard Graduate School of Design | Spring 2018

In collaboration with Thomas Trinelle



The utilization of machine-based recommendation has been leveraged in countless industries, from suggestive search on the web, to photo stock image recommendation. At its core, a recommendation engine can query relevant information -text, images, etc- among vast databases and surface it to the user, as he/she interacts with a given interface. As large 3D data warehouses are being aggregated today, Architecture & Design could benefit from similar practices.

The utilization of machine-based recommendation has been leveraged in countless industries, from suggestive search on the web, to photo stock image recommendation. At its core, a recommendation engine can query relevant information -*text, images, etc*- among vast databases and surface it to the user, as he/she interacts with a given interface. As large 3D data warehouses are being aggregated today, Architecture & Design could benefit from similar practices.

In fact, the design process in our discipline happens mostly through the medium of 3D software (*Rhinoceros 3D, Maya, 3DSmax, AutoCAD*). Might it be through CAD software(*Computer-Aided Design*), or today BIM engines (*Building Information Modeling*), **Architects constantly translate their intention into lines and surfaces in 3D space**. Suggesting relevant 3D objects, taken from exterior data sources, could be a way to enhance their design process.

This is the goal of this article: **study and propose a way to assist designers, through "*suggestive modeling*".** As architects draw in 3D space, an array of machine-learning-based classifiers would be able to search for relevant suggestions and propose alternative, similar or complementary design options.

To that end, taking inspiration from precedents in the field of 3D shape recognition & classification, **we come up with a methodology and a toolset able to suggest models to designers as they draw**. Our goal is, in fact, twofold: **(1) to speed up 3D-modeling process** with pre-modeled suggestions, while **(2) inspiring designers** through alternative or complementary design options.

Convolutional Neural Networks

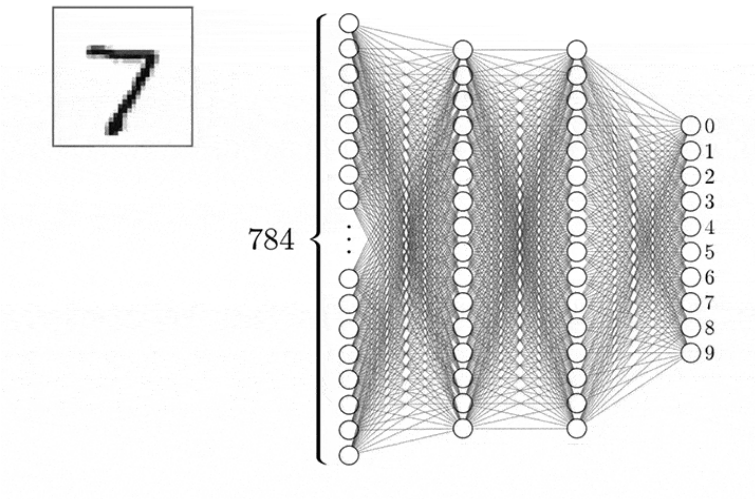


Figure 1: **Convolutional Neural Network Architecture** | Source: 3b1b

Being able to query 3D objects matching the characteristics of a design drawn by the user relies on features comparison. Away from standard geometric descriptive metrics, convolutional neural networks (CNNs) offer a simplified and more comprehensive option to compare shapes.

Instead of matching simple metrics—also called *feature extraction*-, CNNs take images as input, passing a pixel representation to a succession of layers of "*neurons*" (*Figure 1*). A CNN model tunes weights in each neuron while outputting a prediction at its last layer. Through successive phases of training and validation, we are able to estimate the accuracy of our model, and further tune weights to maximize accuracy. Once fully trained, a CNN model will predict the "*class*", or category of a given object image representation.

This type of network is a standard practice in Machine Learning and does not represent a breakthrough or a complex architecture by itself. However, **its ability to build some amount of intuition solely based on spatial features is more than relevant for our application**: the format heterogeneity of publicly available 3D objects makes feature extraction and metadata comparison a challenging process. Being able to consider objects simply from their topological characteristics, using images, offers us a robust unifying framework for 3D-shapes comparison and retrieval.

I. Precedents

Our work builds on top of 3 main research projects, that have recently structured the field of 3D objects recognition and classification. These papers reveal the relevance of convolutions as an ideal tool to understand & describe 3D shapes.

[1] Multi-view Convolutional Neural Networks for 3D Shape Recognition

This first paper **[1]** presents a standard CNN architecture trained to identify the shapes' rendered views independently of each other and shows that a 3D shape can be recognized even from a single view at an accuracy far higher than using state-of-the-art 3D shape descriptors. Recognition rates further increase when multiple views of the shapes are provided.

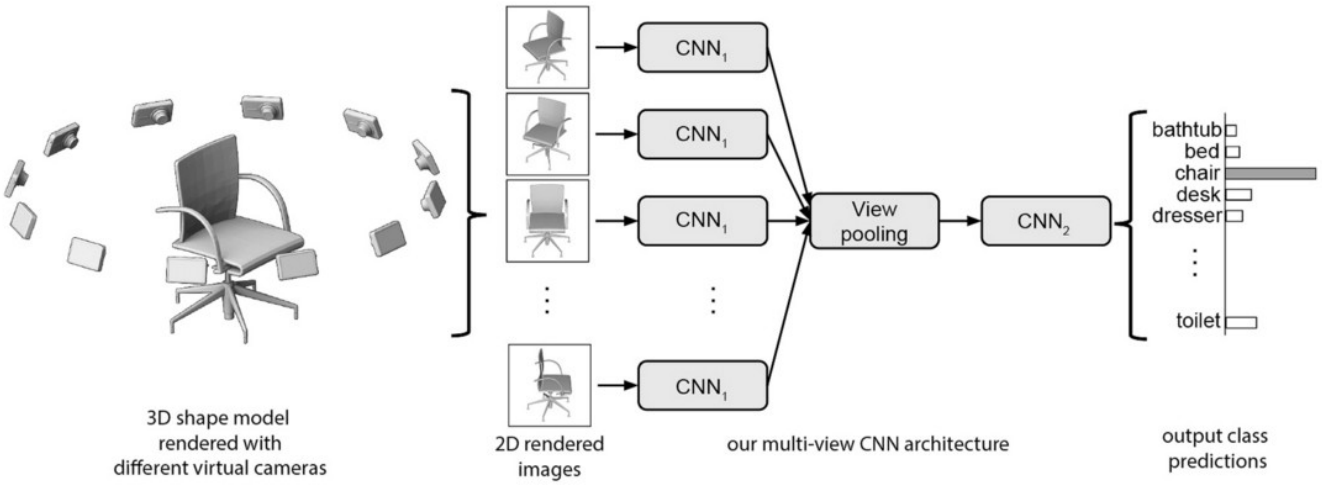


Figure 2: **Multi-view CNN for 3D shape recognition** | Source: Multi-view Convolutional Neural Networks for 3D Shape Recognition

In addition, a novel CNN architecture is introduced, which combines information from multiple views of a 3D shape into a single and compact shape descriptor offering even better recognition performance.

[2] VoxNet: A 3D Convolutional Neural Network for Real-Time Object Recognition

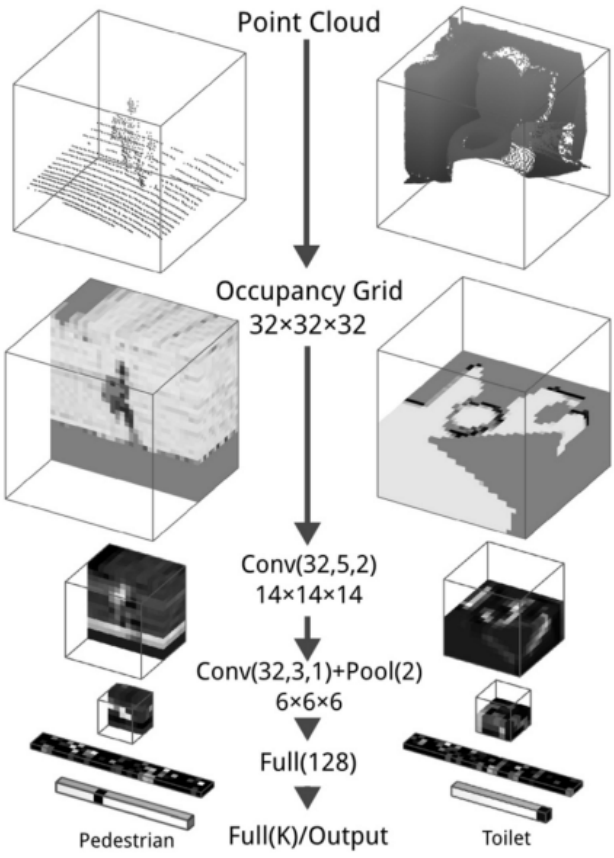


Figure 3: **The VoxNet Architecture** | Source: VoxNet

To leverage the growing amount of point cloud databases, resulting from the increasing availability of LiDar and RGBD scanners, this paper **[2]**, proposes a "VoxNet". This model's architecture aims at tackling the issue of vast point cloud processing and labeling, by integrating a volumetric Occupancy Grid representation with a supervised 3D Convolutional Neural Network (3D CNN).

The results are evaluated on publicly available benchmarks using LiDAR, RGBD, and CAD data. VoxNet ultimately achieves accuracy beyond the state of the art while labeling hundreds of instances per second.

[3] Volumetric Representations And Machine Learning in Design

This last paper **[3]** explores the opportunities of voxel modeling with Machine Learning.

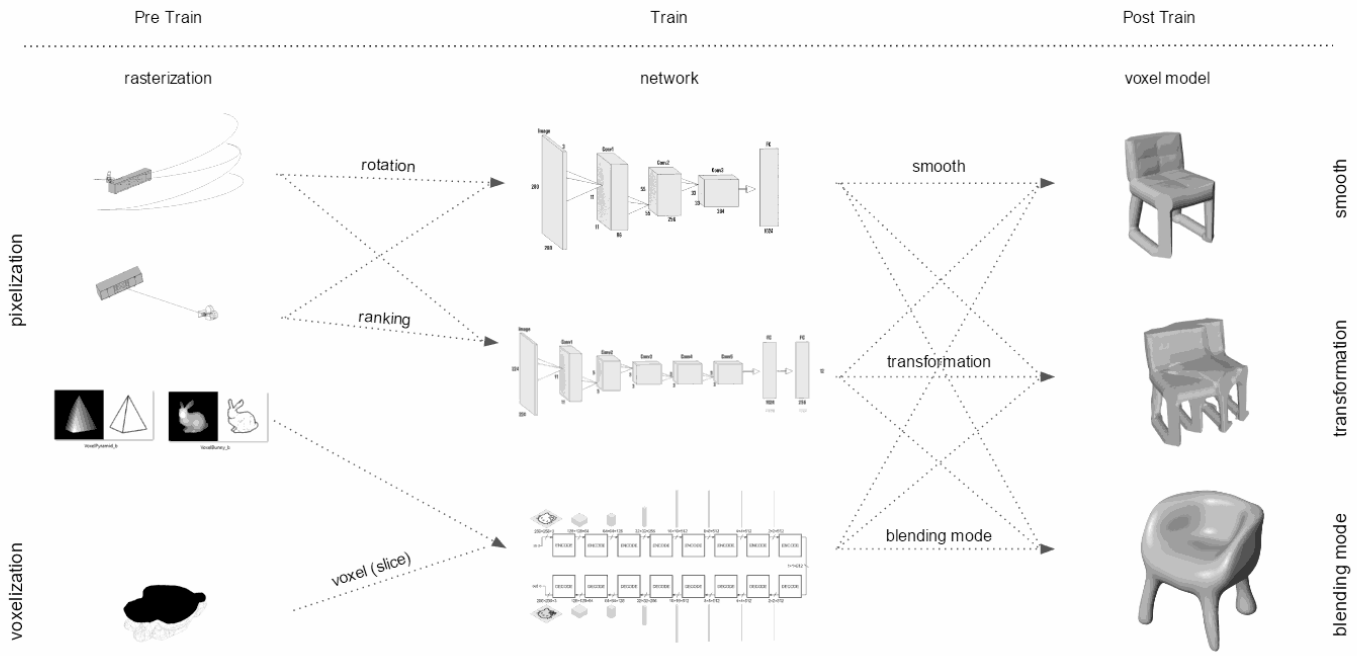


Figure 4: **Summary of Work Flow** | Source: http://njstudio.co.kr/main/project/2017_thesisVoxelHarvardGSD/public/

The notion of voxel modeling is first introduced and compared to traditional model technic. Concepts like pixel map and graph representation are explained, to finally examine prototypical implementations of proposed design systems or workflows based on the process from rasterization of space and geometry with Machine Learning.

II. Model Definition

Our approach in this project is to recognize the object being drawn by the user and to offer similar objects by simply using the shape of the object as a proxy.

In short, the model we are designing here tackles two main tasks:

- **(1) Classifying:** Identifying the type of object being drawn by the user, i.e. finding the proper label (*"chair", "bench", "bed"*, etc.) for any given image input, coupled to prediction confidence score.
- **(2) Matching:** Querying, in a database of 3D-objects, some shapes that best match the aspect of the user's modeled input, i.e. returning a list of objects, found in our database, ordered from most resembling to least.

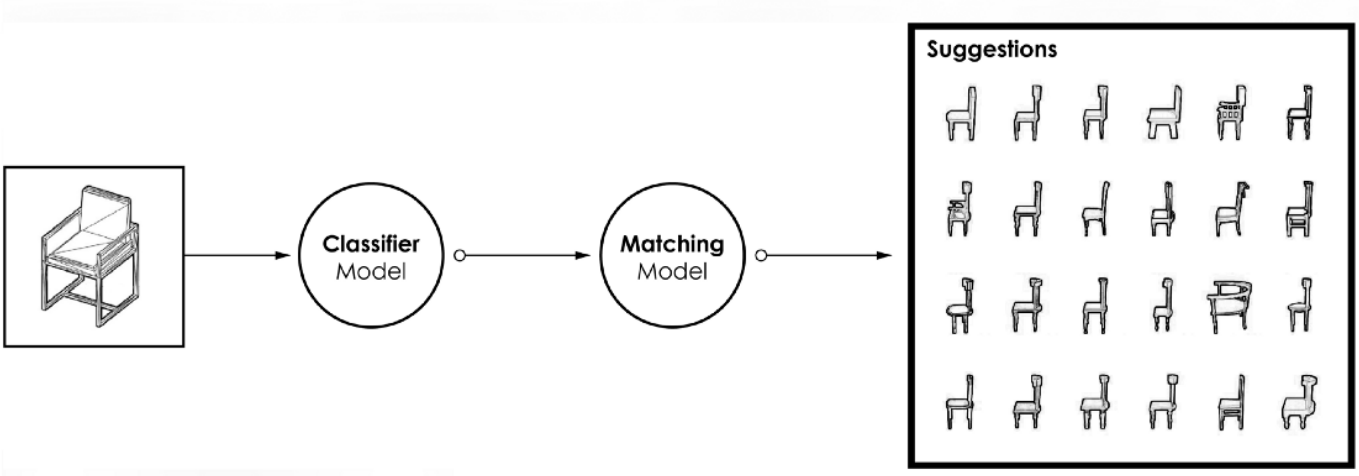


Figure 5: **Typical Pipeline** | Source: Author

As shown in *Figure 5*, by nesting two different levels of models (a **Classifier**and a "**Matcher**"), we might be able to perform this two-step process. Each model will be trained on images of 3D objects and will be then tested on images of objects modeled by the user.

The first step is to generate a database to train our models. **Since our methodology is meant to be shared, we want to detail here this crucial step and share the resources we used and built to achieve it.**

We first extract information from existing public 3D objects warehouses such as:

- **ShapeNet**
- **Google 3D Warehouse**
- **ModelNet**

From the **ShapeNet** database, we can download up to 2.330 labeled 3D models, of 14 specific classes (*Figure 6*).

- **Sink** (120 objects)
- **Sofa** (200 objects)
- **Stairs** (120 objects
- **Table** (200 objects)
- **Toilet** (200 objects)
- **Bathtub** (100 objects)
- **Bed** (200 objects)
- **Bench** (170 objects)
- **Bookshelf** (200 objects)
- **Chair** (200 objects)
- **Door** (100 objects)
- **Lamp** (120 objects)
- **Plant** (200 objects)
- **Vase** (200 objects)

Figure 6: **Training Set Classes** | Source: Author

Using *Rhinoceros* and *Grasshopper*, we then script a tool to create both our training and validation sets. In this script, a camera, rotating around each successive object takes shots of the object at specific angles and saves JPG images in a given directory.

Figure 7 illustrates a typical camera path (left), and the resulting images captured (right).

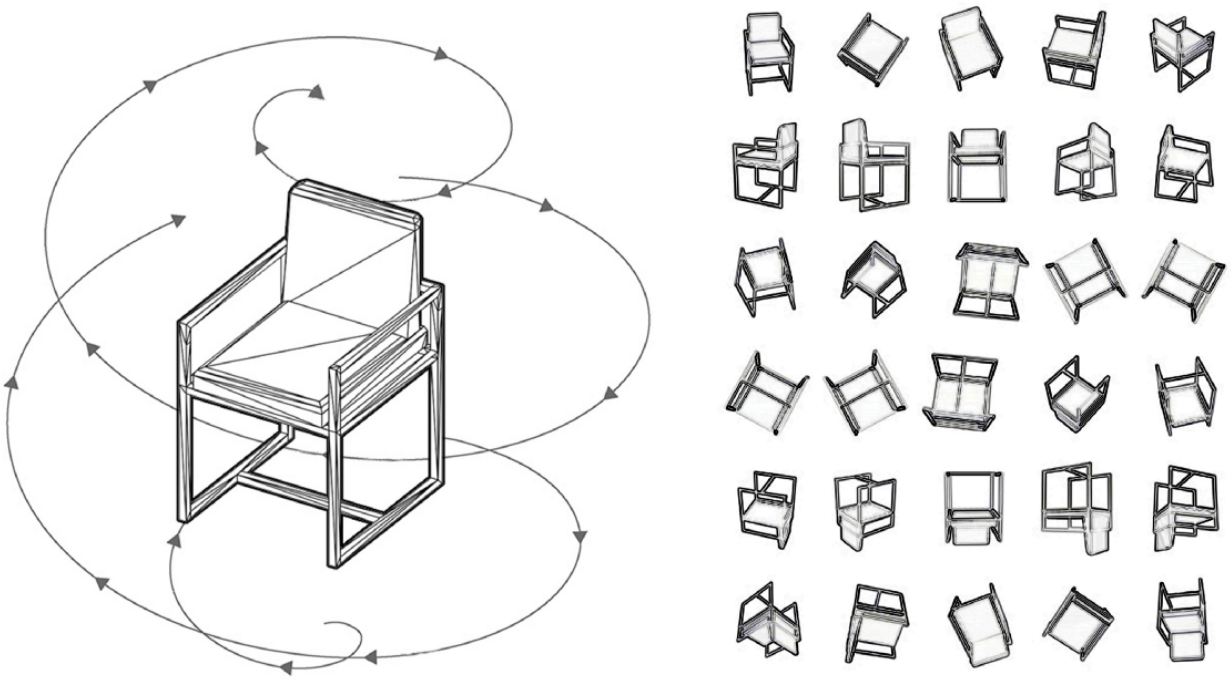


Figure 7: **Image Capture Path, and Resulting Images** | Source: Author

For each object, we take 30 images for training, and 10 for validation, while keeping a white background/neutral background.

We finally obtain a library of labeled images, for 10 objects per class, having a total of 14 different classes. A subset is here shown, in *Figure 8*.

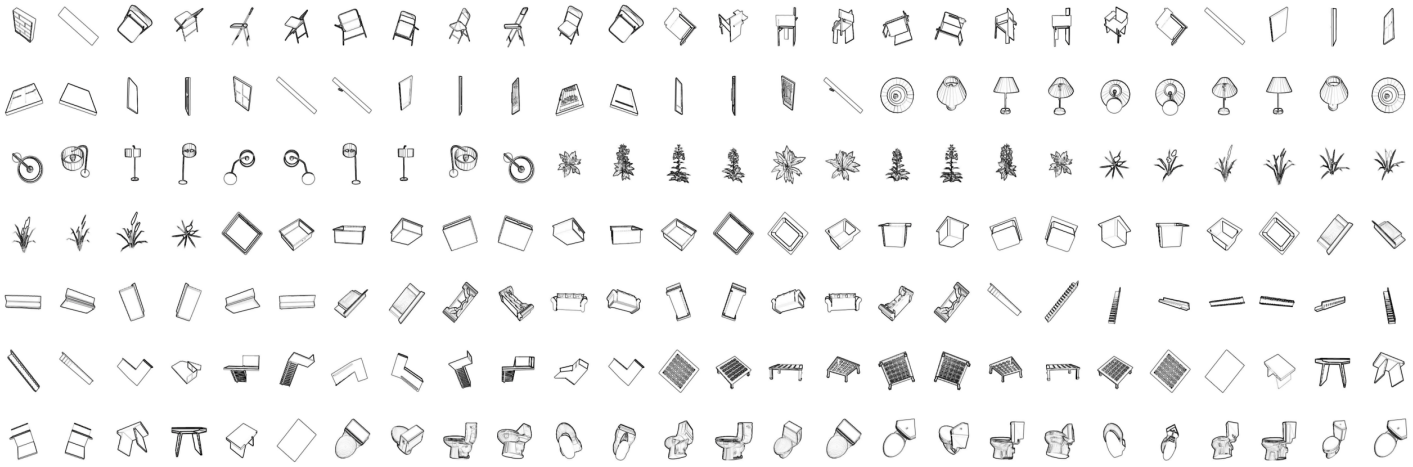


Figure 8: **Subset of Training Set** | Source: Author

B. Classification

Once our dataset is ready, our goal is to train our first model, the classifier, on extensive amounts of 3D objects' images, while validating its performance on sets of other objects' images, belonging to similar classes.

Iterations

Several parameters heavily impact the accuracy of the trained classifier:

- **The size** of the training and validation set
- **The number of classes**
- **Number of object**s per classes
- **Number of images** per object
- **The size of the image**s in both sets
- **The capture path** of the camera around the object

At this point, we iterate between different options and aim at increasing the model overall accuracy on the validation set. Having limited resources, we cannot afford to train from scratch. **Using some transfer learning comes in handy, as it offers the possibility of increasing our model accuracy while bypassing days of training**. We add as first layers of our model a VGG16 pre-trained model, which boosts in return our accuracy by 32%.

A key takeaway of this training process is that the camera path actually impacts significantly the resulting accuracy. Among the many versions we have tried, we illustrate here below, in *Figure 9*, the comparative performance of two different camera path: **circular** and **spherical**.

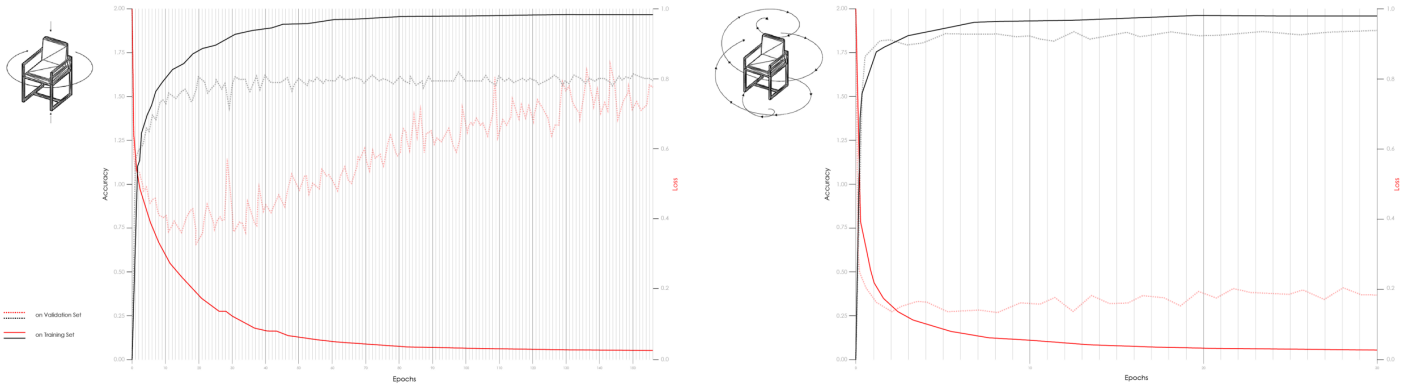


Figure 9: **Training Performance Under Different Image Capture Technics** | Source: Author

Result

Ultimately, we settle for a classifier using *200*200* px images, with a spherical camera path, 30 images per object for training, and 10 for validation. After 30 epochs, **we finally obtain an accuracy on the validation set of 93%.** It seems clear now that the following parameters influence greatly the performance of the overall model:

- **Increasing the image size increases accuracy.** On our machine, the trade-off between speed and accuracy finds its balance around *200x200* px .
- **The camera capture path directly also impacts the accuracy.** By using a spherical capture path, instead of a circular path (*see Figure 9*), we boost significantly our model performance: the accuracy is way higher, after fewer epochs. **Using a spherical path simply seems to be a more holistic way to capture a given shape's aspect.**

Figure 10 shows typical results of the classifier, for four different user inputs.

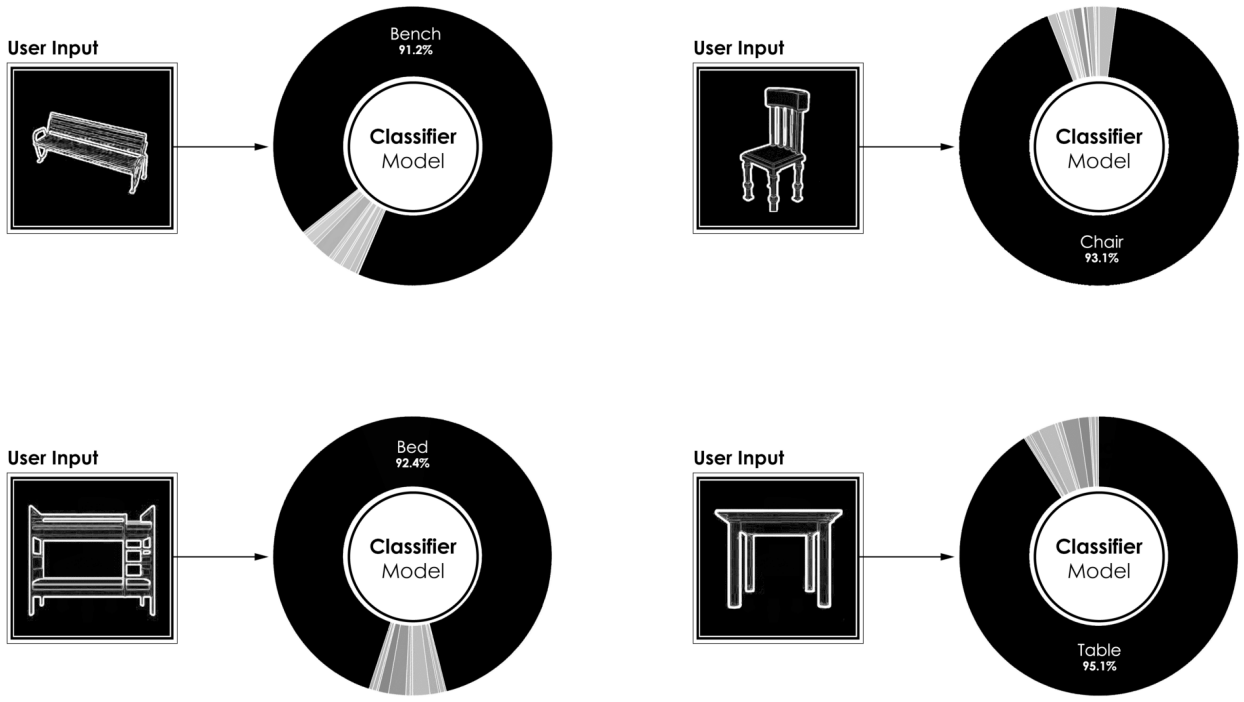


Figure 10: **Results of Classifier Model, User Input (left) to Predicted Class (right)** | Source: Author

- **More interesting even is the performance of the same model, ran during the 3D-modeling process of a given user.** As an object gets modeled, as little as few surfaces will suffice to nudge the classifier in the right direction. As a result, early on in the modeling process, we are able to identify the class of the modeled object. In return we can suggest models from our database to the user, potentially reducing the 3D-modeling time, if the user finds a match among the suggestions. *Figure 11* displays the classifier's results, at five different steps of the modeling process.

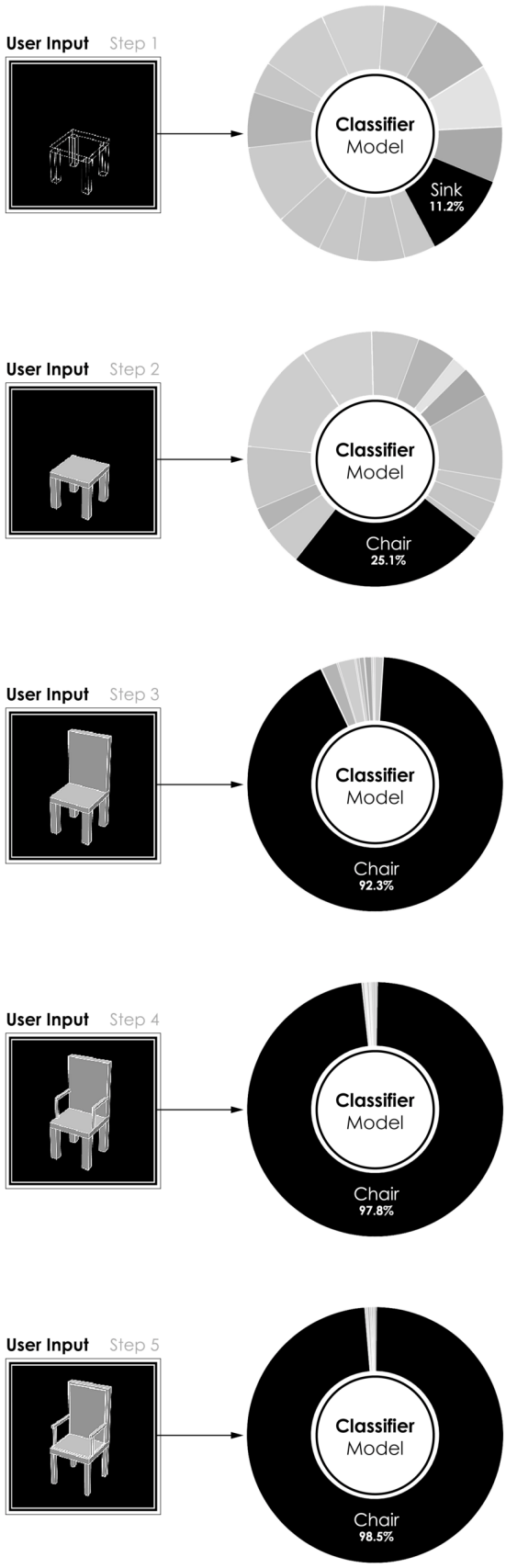


Figure 11: **Classification During Modeling Process** | Source: Author

C. Matching

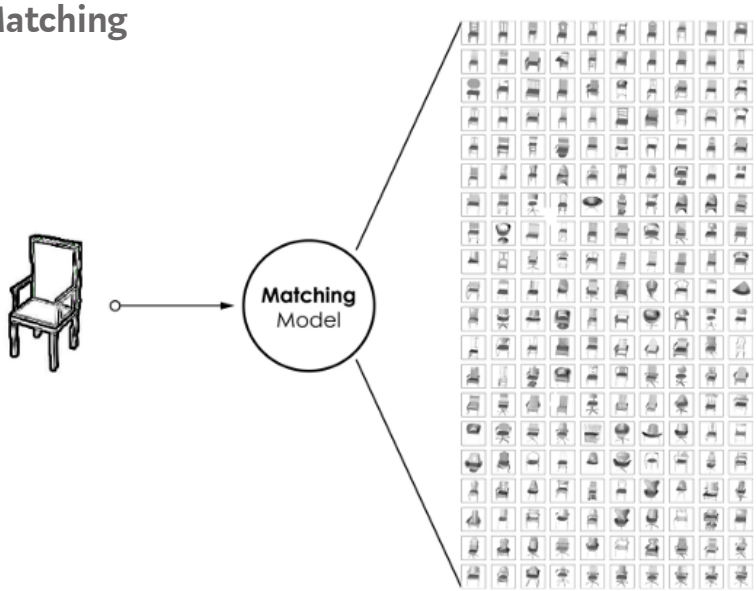


Figure 12: **Matching: Finding Best Matches Among Database Through a Convolutional Model** | Source: Author

In a second step, **our model attempts at finding ideal matches in an extensive database of 3D models**. If the classification model helps in narrowing down the search, the matching model ranks all 3D models from a given class from “most resembling” to “least”.

Our matching model is another convolutional neural network, trained on images of objects of a given class, and validated on the same objects, with views taken from different angles. For testing, our input will be one image of a user-modeled 3D object, and the output will be a list of all objects in our database of the identified class, ranked in order of resemblance.

Result

As we train a matching model for each individual class, we are now able to nest the classification model and the matching models as one single pipeline. **We can now process an input image, that will be first classified, and finally matched with similar objects present in our database.** At the same time, our model outputs some prediction confidence to help us gauge how strong the resemblance between the original model and the match actually is.

Tests have been run for 7 objects of different classes and are displayed in Figure 13.

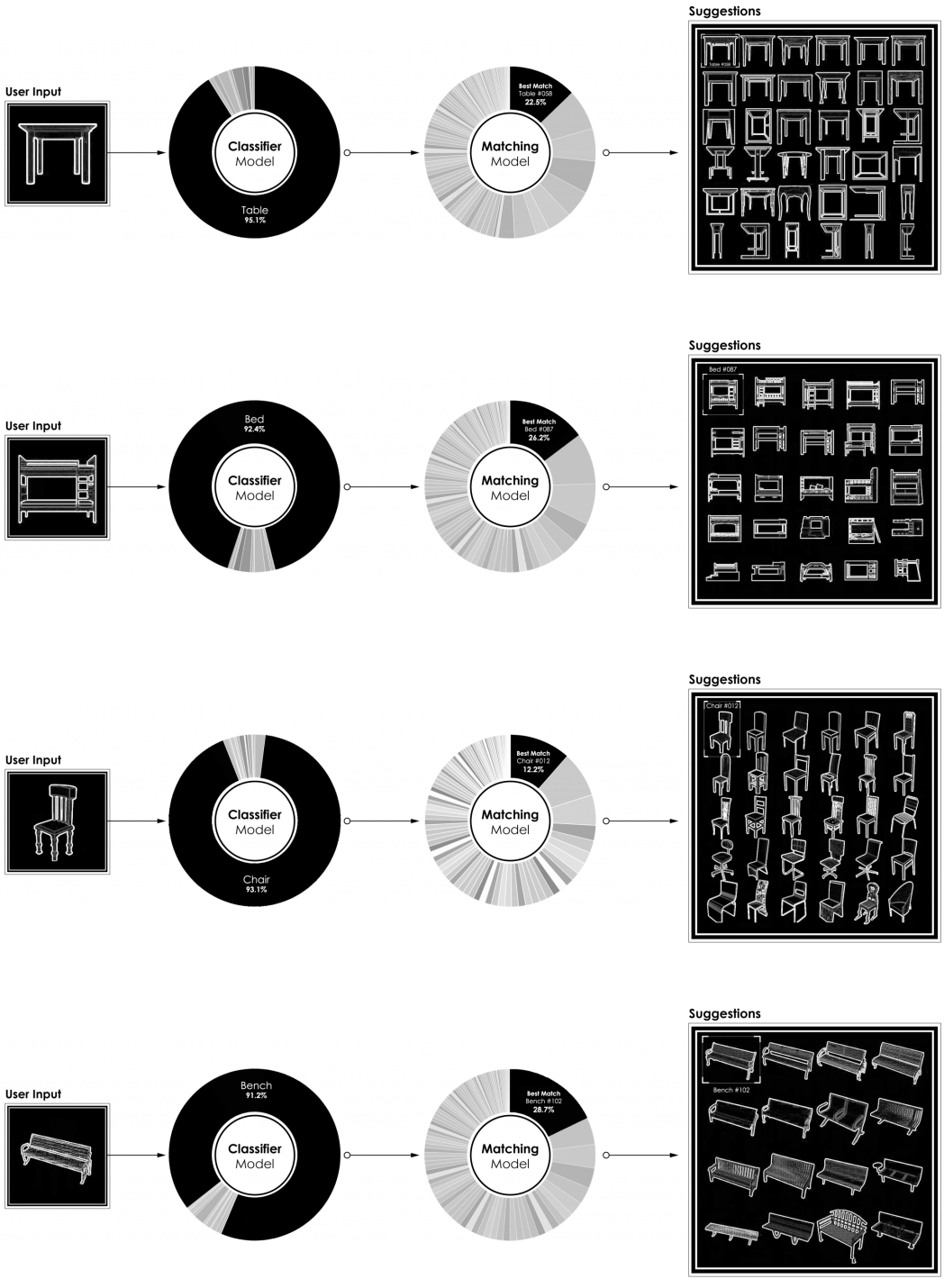


Figure 13: **Full Pipeline With Functional Classifying & Matching Model** | Source: Author

III. Conclusion

First, we encourage further developments and improvements of the above idea, within our industry. A lot has been done in this area and has brought solutions to adjacent fields. Allowing these technologies to permeate in our discipline would be truly beneficial to our daily practice of Architecture. Also, nesting CNN models as we did is only one possible version of a larger concept: **3D-shape recognition and suggestion**. Other approaches and models could be deployed to bring the breadth & richness of publicly available data within reach of architects & designers.

In fact, beyond the simple pipeline developed in this article, lies a more fundamental idea: **the qualification of architectural forms**. As described in a previous article, being able to structure frameworks, encompassing the heterogeneity and complexity of existing shapes, will be soon crucial to our discipline. As the amount of digital data increases and gets pooled together in large public repositories, **our access to this shared knowledge will only be as good as the intelligence of our queries. As we demonstrated here**, we can rely *-to some extent-* on Machine Learning to find a common language, enabling the comparison of diverse & complex shapes.

At a more fundamental level, this work simply shows the potential of suggestive design. By bringing relevant alternatives to designers during the creative process, we have the opportunity to broaden the scope of their work. As this approach scales to taxonomies larger than simple discrete objects, it ultimately would turn passive collective knowledge into an active source of inspiration.

Bibliography

[1] Multi-view Convolutional Neural Networks for 3D Shape Recognition, Hang Su, Subhransu Maji, Evangelos Kalogerakis, Erik Learned-Miller, University of Massachusetts, Amherst

[2] VoxNet: A 3D Convolutional Neural Network for Real-Time Object Recognition, Daniel Maturana and Sebastian Scherer

[3] Remixing and Resampling Three Dimensional Objects Use of Volumetric Representations And Machine Learning in Design, NJ Namju Lee, Harvard GSD, 2017