A Practical Use of Machine Learning in The AEC Industry

John D’Alessandro and Doug Johansing

Software Engineers

AUTODESK
UNIVERSITY

Agenda

INTRODUCTION – 5 MINS

CASE STUDY – 30 MINS

Background, an overview of the layer name translator, our practical ML solution, and the problem that it solves.

LESSONS LEARNED – 15 MINS

Some takeaways from our experience with using machine learning that you all can bring home with you.

From This Presentation, You Will…

DISCOVER HOW MACHINE LEARNING CAN BE USED AS A TOOL IN THE AEC INDUSTRY

We will cover our Layer Name Translator, a machine learning solution for a problem KLH Engineers faces daily.

DISCERN WHAT PROBLEMS CAN BE SOLVED USING MACHINE LEARNING

We will cover why this problem was a good fit for machine learning.

LEARN HOW TO INTEGRATE A CONCRETE USE OF MACHINE LEARNING INTO A COMPANY’S WORKFLOW

We will cover what speed bumps we hit along the way to being able to use this new technology consistently across our company.

John D’Alessandro

Software Engineer, KLH Engineers

Since entering the AEC industry, John D’Alessandro has developed several innovative tools within and outside of Revit to help engineers spend less time drafting and more time engineering. He has also developed machine learning tools on Azure using .NET and Python. John holds a Bachelor of Science in computer science from the University of Cincinnati.

Doug Johansing

Principal | Director of Software Engineering

KLH Engineers

Doug Johansing, PE, LEED AP BD+C, is a principal of KLH and serves as a programmer and senior electrical engineer. He has over 15 years of experience in the AEC industry and 10 years working with the Revit and AutoCAD APIs. He leads many database driven programming initiatives related to Revit and process innovation. Doug holds a Bachelor of Science in electrical engineering from the University of Cincinnati.

Background

Background

KLH Engineers made the transition to utilize Revit as the only design tool while many clients continue to use AutoCAD.

As a national MEP engineering firm headquartered in Greater Cincinnati, KLH's clients are:

Architects

Contractors

Civil Engineers

Background

KLH Engineers made the transition to utilize Revit as the only design tool while many clients continue to use AutoCAD.

For years, KLH maintained process workflows and programming efforts for both AutoCAD-based clients and Revit-based clients. A handful of engineers at KLH were both engineers, and programmers, which pushed the envelope of the AutoCAD and Revit APIs.

In 2016, knowing that central databases and BIM-based models were a necessity for the future of the industry, KLH had to stop supporting AutoCAD-based engineering and design workflows.

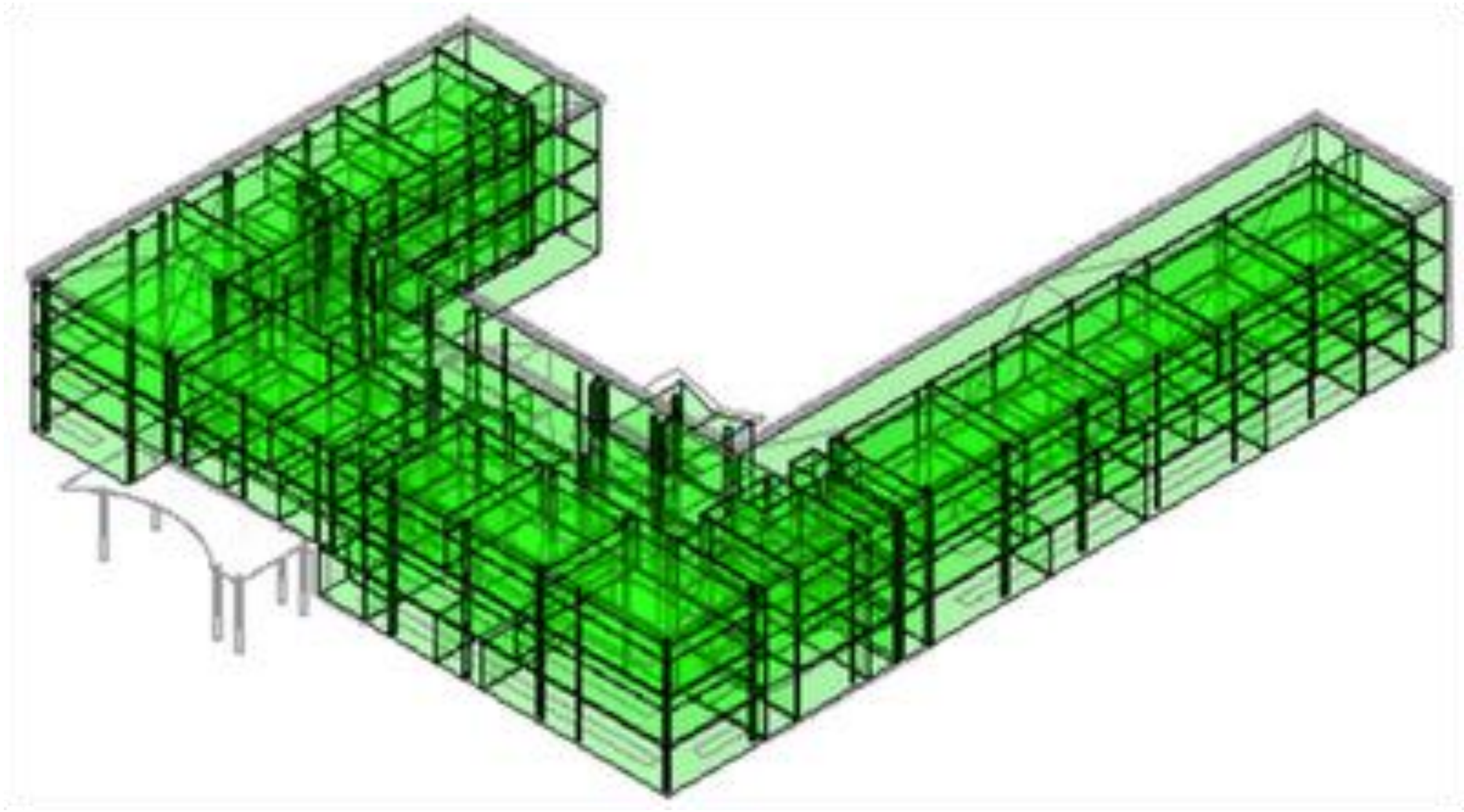
So, the 100% Revit initiative began …

Background

KLH Engineers made the transition to utilize Revit as the only design tool while many clients continue to use AutoCAD.

KLH focused its efforts to migrate tools and processes into the Revit environment …

- New data centric efficiency tools were developed that required:
  - Spatial Information
  - Parameters
  - External Database Integration

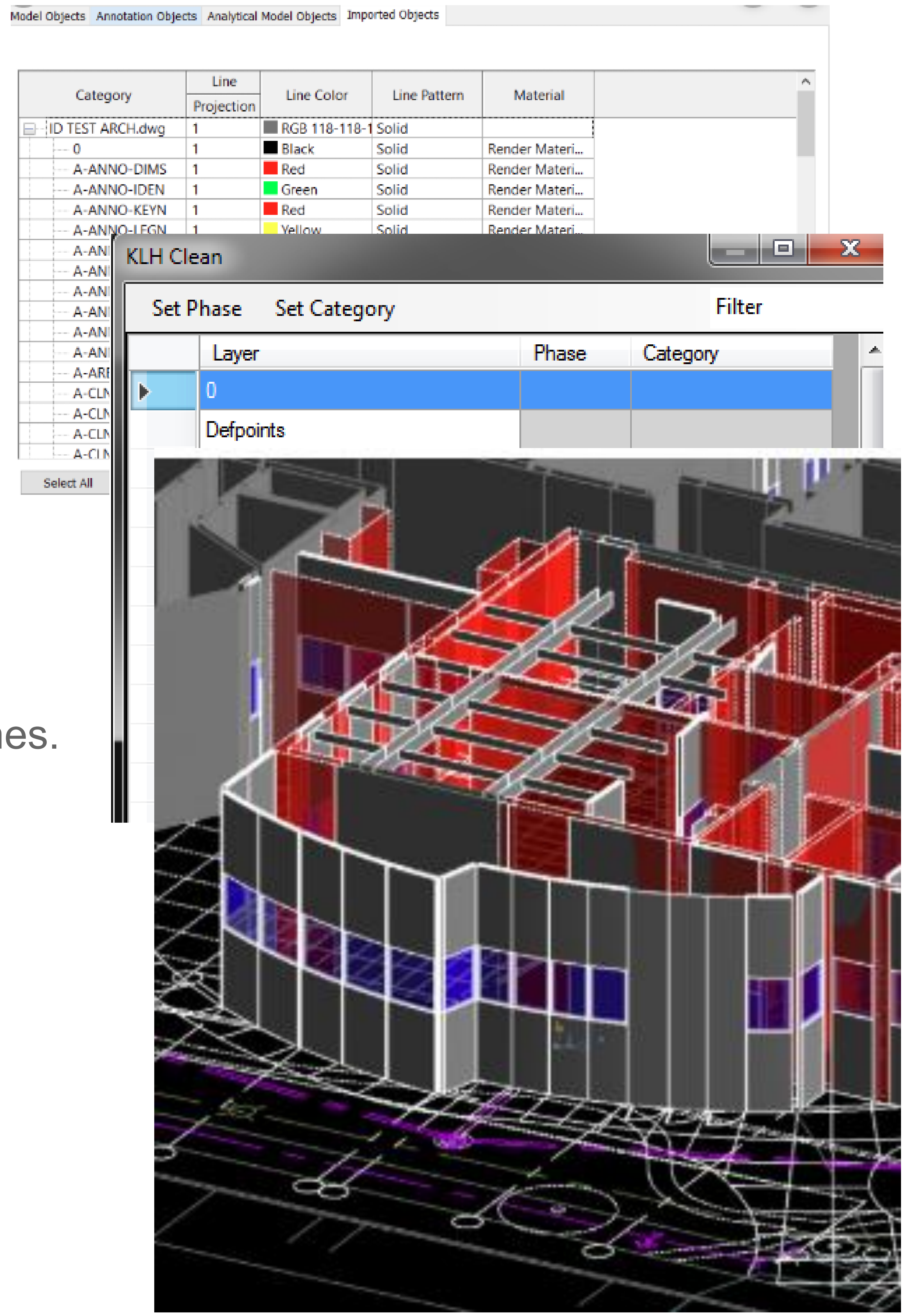


Background

KLH Engineers made the transition to utilize Revit as the only design tool while many clients continue to use AutoCAD.

AutoCAD was still one of the main data formats KLH was receiving … and it wasn’t efficient.

- Revit has a built in AutoCAD file format linking, but managing the “Import” visibility settings would be too cumbersome.



So, KLH created two tools:

1. A tool to map client layers into standard KLH layers and perform AutoCAD “cleanup” type routines.
   - KLH already had a manual .dwg “cleanup” process that saved a .csv of different layer mappings per job.
2. A tool to “extrude” the AutoCAD lines as native Revit elements (walls, model lines, etc.) so our Revit tools could be fully utilized when AutoCAD is the communication format.
   - Native 3D elements created boundaries, which allowed for spaces / spatial information.
   - Native elements were associated with a Revit phase, which allowed for phase filtering.

The Layer Name Translator

CAD Conversion

STAGE A

GET CAD MODEL

We receive an unstandardized 2D CAD file from an architect

STAGE B

CLEAN MODEL

We clean the 2D model manually, it now conforms to our standards

STAGE C

CONVERT MODEL

With the standardized model, our Revit addin can convert the 2D lines in the CAD model into a 3D Revit model.

CAD Conversion – Manual Cleaning

STAGE A

SELECT LAYER

Using our cleaning tool, the designer will select one layer at a time to evaluate its standard designation

STAGE B

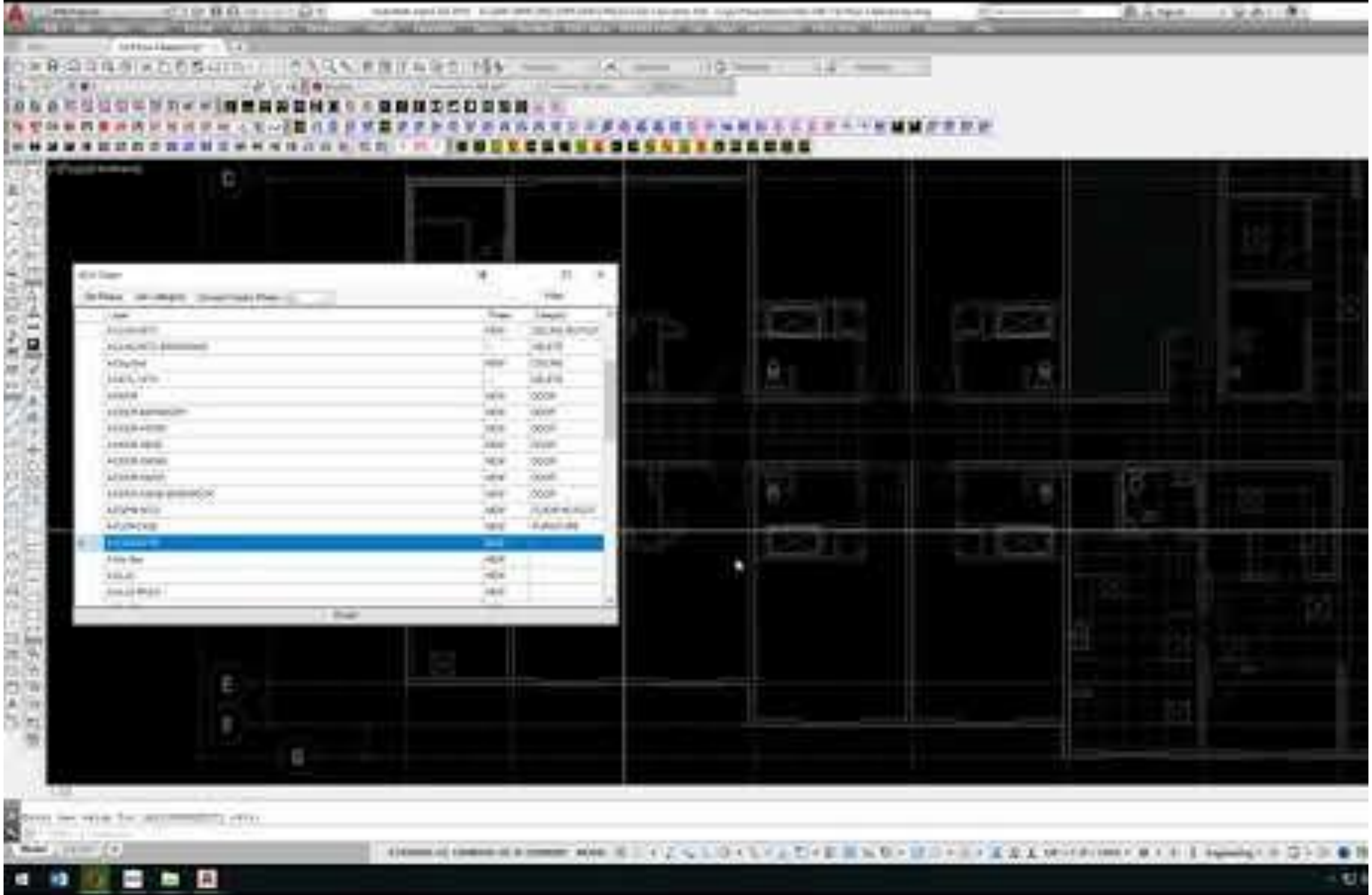
ASSIGN PHASE

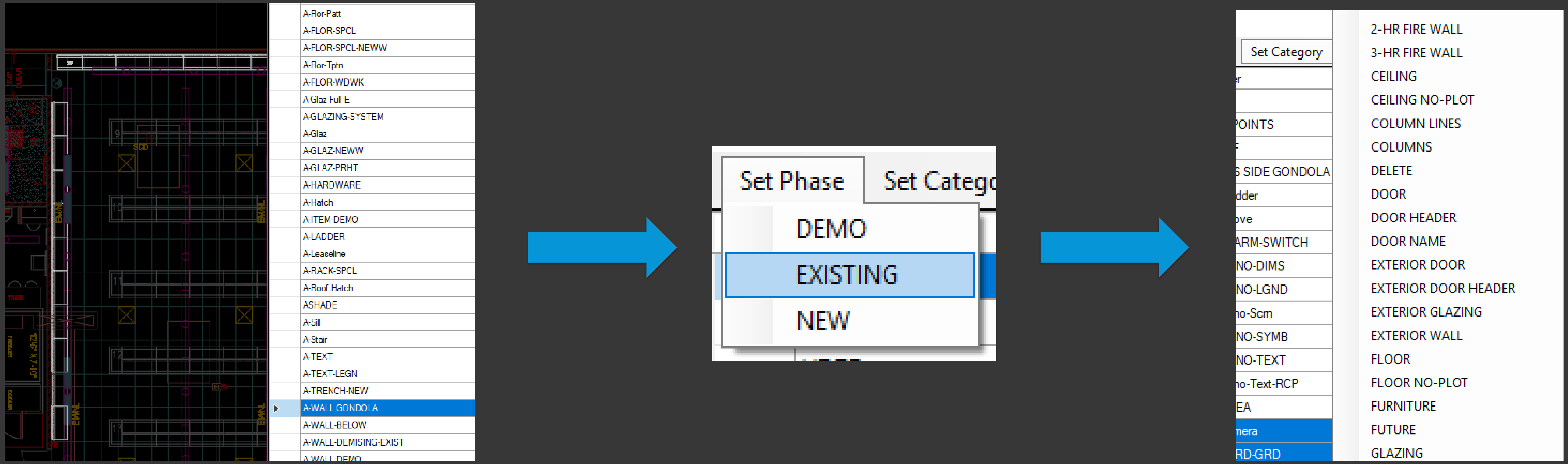
The designer must determine the phase of the architect’s layer.

STAGE C

ASSIGN CATEGORY

The designer must determine the category of the given layer, which eventually tells our Revit tool how to translate the CAD layer into Revit objects





Problem

The manual method of cleaning AutoCAD files for conversion takes a large amount of manual time by skilled designers, and encourages them to take shortcuts that may lead to issues with conversion.

A Possible Solution

A generative approach was considered. Rules would be developed to take architect layer names and translate them into our standardized categories.

- This would work for the designers if implemented properly
- Problems arise
  - There are boundless edge cases that need to be considered
  - The rules would need to be constantly amended
  - The algorithm as a whole would be impractical to maintain
- We determined that we needed a better solution

```
Public Function DetermineLayer(s As String) As String
    With s.ToUpper()
        If .Contains("DOOR") Then
            Return "N-AR-DOOR"
        ElseIf .Contains("WALL") Then
            Return "N-AR-WALL"
        ElseIf .Contains("FURN") Then
            Return "N-AR-FURNITURE"
        ElseIf .Contains("FURNITURE") Then
            Return "N-AR-FURNITURE"
        Else
            Return "DELETE"
        End If
    End With
End Function
```

	A	B
1	P-PLUMB-EXIST	X-AR-PFIX
2	S-COL-TAG	X-AR-COLL
3	Ceiling Tag	X-AR-CNPL
4	A-FLOR-CASE	N-AR-FURN
5	I-FUR	N-AR-FURN
6	YE-pnts-topo-spot-ttoe	DELETE
7	A-Staircase Railing - Ex'g	X-AR-STRS
8	S-SYMB	DELETE
9	A-EQPM-PRHT	DELETE
10	Existing Lighting	D-AR-LFXT
• • •		
38592	N-AR-WALL-N	N-AR-CNPL
38593	Shell0A-SHEL-INTR-01NFLR	X-AR-WALL
38594	F-BOH DTL	X-AR-FURN
38595	A-DEMO	D-AR-WALL
38596	24x36 Border Bassett Frisco-CDS0	DELETE
38597	I-CLNG-OPEN	N-AR-CNPL
38598	A-ANNO-TEXT	DELETE

Layer	Phase	Category
0		
DEFPOINTS		
XREF		
18x36 SIDE GONDOLA	NEW	FURNITURE
A -Ladder	NEW	FURNITURE
A-Above	--	DELETE
A-ALARM-SWITCH	NEW	FURNITURE
A-ANNO-DIMS	--	DELETE
A-ANNO-LGND	NEW	ROOM NAME
A-Anno-Scrn	--	DELETE
A-ANNO-SYMB	--	DELETE
A-ANNO-TEXT	--	DELETE
A-Anno-Text-RCP	NEW	CEILING NO-PLOT
A-AREA	--	DELETE
A-Camera	--	DELETE
A-CARD-GRD	NEW	FURNITURE
A-CART-GRD	NEW	FURNITURE
A-Clng Fixt Above	NEW	CEILING
A-CLNG-GRID	NEW	CEILING

Machine Learning

With 38,000 decisions already made manually by our designers, we were able to train a model to look at the architect’s layer name and make a decision that a human would almost instantly. In addition, the model did not take shortcuts and gave more accurate classifications than many of the people were able to manually.

Programming Effort

The machine learning part of the solution was not difficult. It only took 130 lines of Python to train the model, and 250 lines to make the server that our AutoCAD addin interacts with.

- Tensorflow and Flask, the technologies we used, are both free
- The programming effort was minimal

The difficult part of this process was getting the data and meaningfully quantifying it.

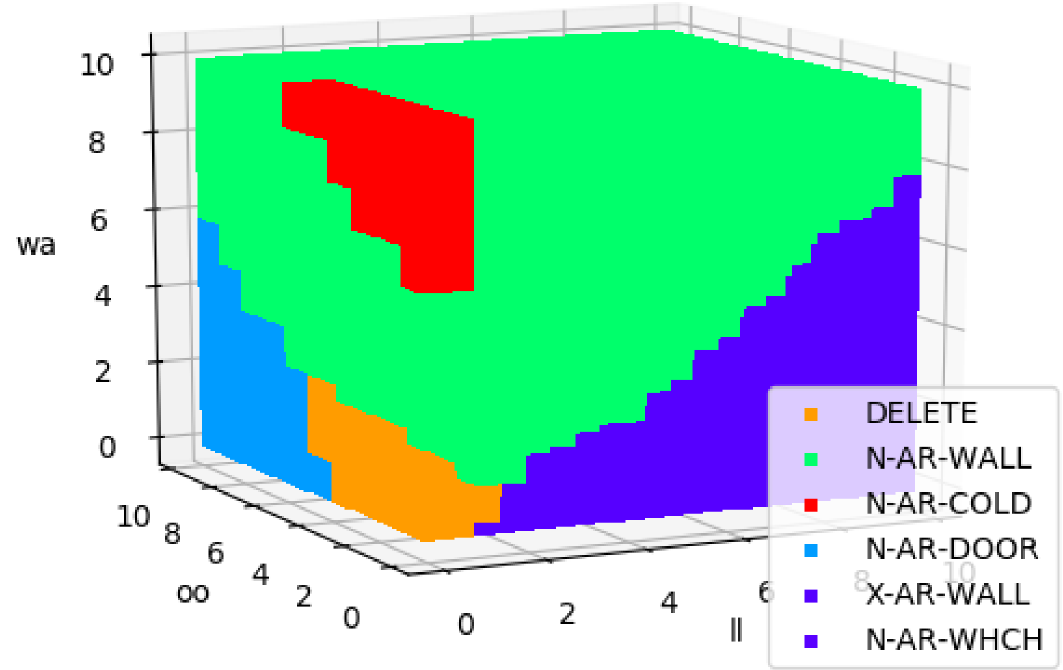
Technobabble

Our algorithm is a simple text classification one using a softmax algorithm that gives us a confidence in each of our 130 layer categories. Our input is a bigram built from the string. Essentially a point in 1369-dimensional space where each dimension represents the frequency of a given pair of letters. The machine learning model is simply a division of this space into regions that map to our categories

NEW WALL

```
def to_2_gram(string:str) -> []:
    two_gram_array = get_two_gram_vector()
    ret_array = [0] * len(two_gram_array)
    for i in range(len(string) - 1):
        try:
            index = two_gram_array.index(string.upper()[i:i+2])
            ret_array[index] = ret_array[index] + 1
        except:
            continue
    return ret_array
```

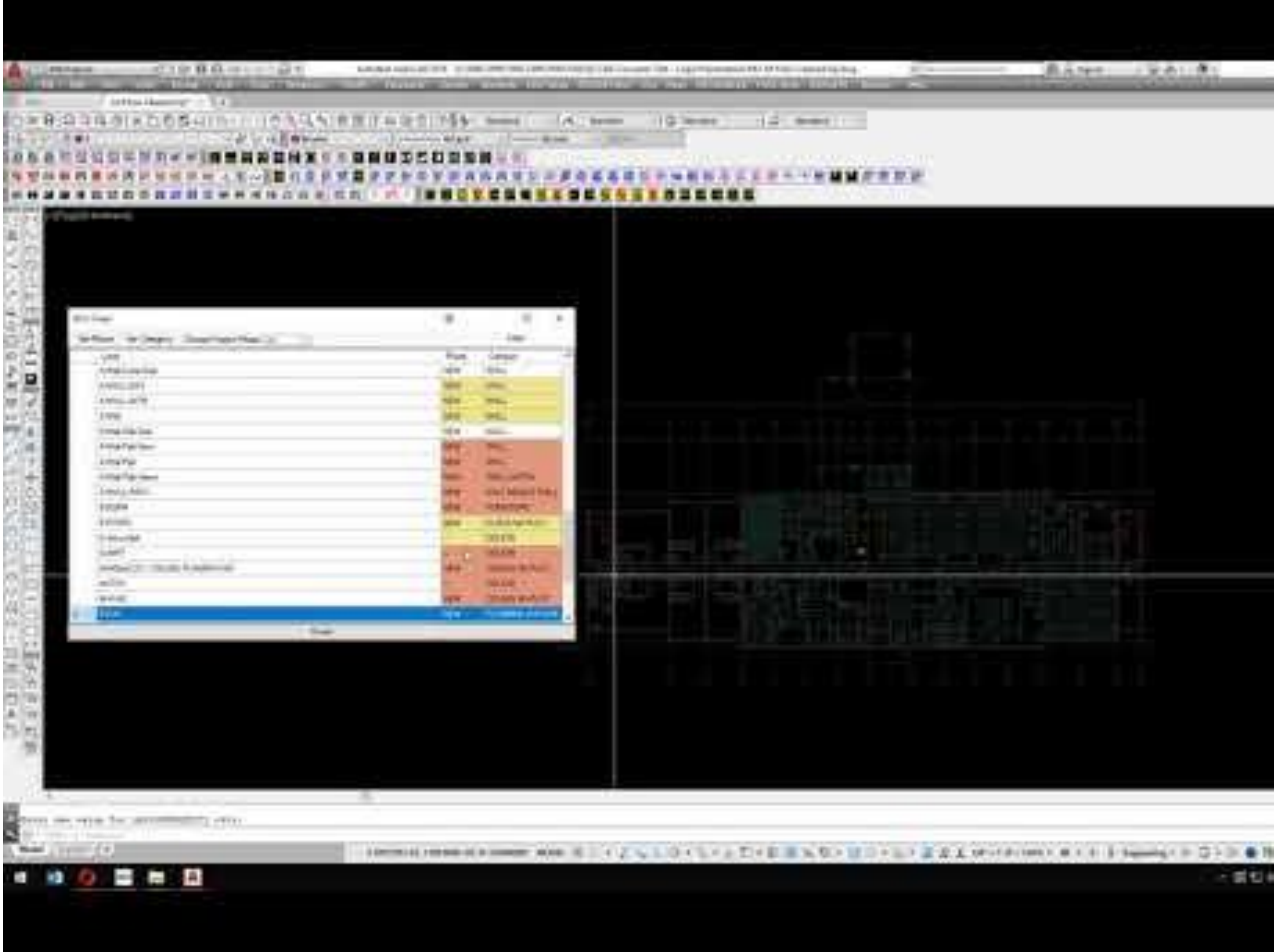
(AA,AB,…,AL,…,EW,…,???)
(0 , 0 ,…, 1 ,…, 1 ,…, 0)



N-AR-WALL (49% confidence)
X-AR-WALL (39% confidence)
N-AR-WHCH (3% Confidence)
X-AR-WHCH (1% Confidence)

Advantage

The machine learning method allows us to train a model for about ten minutes and then quickly classify any layer name we could possibly get.



Lessons Learned

Why Machine Learning Worked

EXISTING DATA

KLH’s tools had already been tracking the data about decisions that designers were making. The 38,000 rows of data was instrumental in the development of a solution driven by previous decisions. Without the wealth of history at our disposal, the model would have been inaccurate and useless.

CLEAR INPUTS AND OUTPUTS

We set out with a clear and precise definition of what our model would look at and what it would spit out. This meant that we could analyze the data easily and measure the effectiveness of the model. Without a defined problem to solve, no data science could give a satisfactory solution

PATTERNS EXIST EVEN THOUGH THEY AREN’T CLEAR

Though there is no explicit standard for how architects name their AutoCAD layers, there are patterns that a model can learn from. If it sees an “LL” in the layer name, it is more confident that that layer represents a wall, because historically architects put “LL”s in the names of their wall layers. Without these patterns, any model we developed would just be guessing.

What Went Wrong Along The Way

DATA ANALYTICS

Before the bigram algorithm, we were trying to feed the model an array of integers representing the characters (so "WALL" would look like (87, 65, 76, 76, 0, 0, …). The model couldn't find patterns because we weren't representing the data in the same way that humans read it: in sounds formed by sets of letters.

USER EXPERIENCE

Early in the development of the tool, the machine just filled out the list of layer names for the designer. This made it opaque to the designer that there was a machine assisting with design decisions. We added colors to show the confidence of the machine in each decision to guide the human in verifying and correcting those decisions.

DATA VALIDATION

We thought the model was doing very poorly, with about an 80% accuracy, until people actually started using it. Because the model was more accurate than the designers, the model's correct decisions were being scored using incorrect ones, giving us a lower accuracy number.

Advice

KNOW WHAT YOU WANT

Don’t use a technology for its own sake. Apply tools to the problems they are designed to solve. Machine learning is a tool that is good for a specific set of problems. Keep it in your toolbox, but use whatever works best for your desired process.

KEEP ALL OF YOUR DATA

This project would have been impossible if it weren’t for our data we were already collecting. The data can be massaged and formatted down the line when a use for it comes up, but it will be much easier to store data now than to generate it later for that use.

USE WHAT’S OUT THERE

Unless you’re an academic, all of the hard parts of machine learning have been done for you. Find tools and information related to whatever problem you’re trying to solve. We used free technologies and techniques we found through research to get what we needed. Don’t overcomplicate your problem by reinventing the wheel.

AUTODESK®

Make anything™