AI-Based Webapp for Microfluidic Design Automation

Code Review

Group 63: Isaac Jensen, Joe Lei, Qiuhong (Barret) Li

Group Introduction

- Isaac Jensen: Oregon State Senior (Developer/Team lead)
- Joe Lei: Oregon State Senior (Developer)
- Qiuhong (Barret) Li: Oregon State Senior (Developer)
- HP R&D Engineering Modeling and Analysis Group (EMAG)
  - Project Partner: Jacob Lum, Matt Sundheim, Mandana Hamidi Haines, Anand Jebakumar

Project Overview and Context

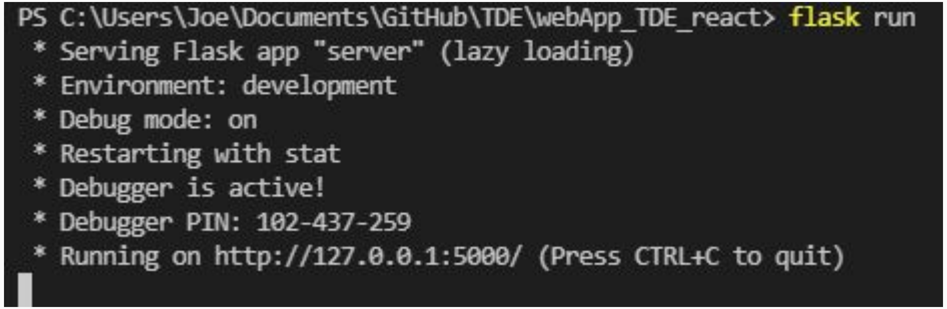
- We were tasked with creating a webapp for the HP R&D EMAG team in order to help with their prediction process. Our webapp lets the user input fluid parameters, create fresh .GEO files (layer based), and run them through our PyTorch neural network to gain valuable statistics about the geometry.

- Our goal for this project is to build an interactive interface to output the parameters for Thermal Inkjet printhead performance. The simulation operates based on a deep neural network built on a large historic fluidic database. These predicted outcomes will help engineers adjust their design for the Thermal Inkjet printhead to meet the expected performance. This will save huge amounts of time and money in experimenting with different designs.

Our Build

- Python (Flask) server running our webapp
- Node.js
- PyTorch (Neural Network)
- .GEO file construction scripts
- HTML forms for input/output
- Plotly.js (graphing library)
- Prediction Profiler (graphing tool)
- Jinja2 (inside Flask, used for templating)
- Webpack (JS module bundler)

Flask Integration

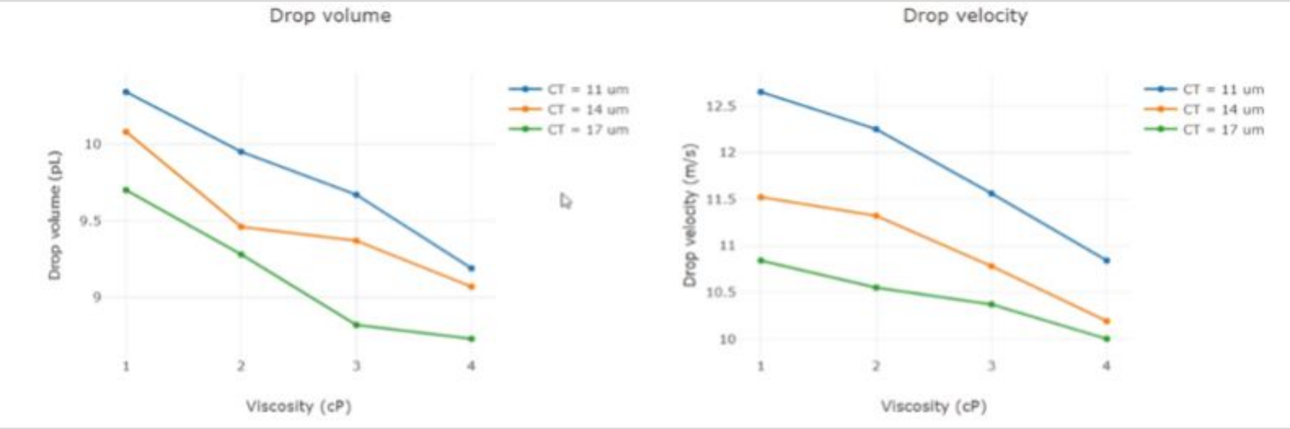
- Used Flask as our main web application framework.
- We used python Flask because it was easy to integrate and work with and easier to route the given code by our project partners.
- 

```
PS C:\Users\Joe\Documents\GitHub\TDE\webApp_TDE_react> flask run
 * Serving Flask app "server" (lazy loading)
 * Environment: development
 * Debug mode: on
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 102-437-259
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

PyTorch

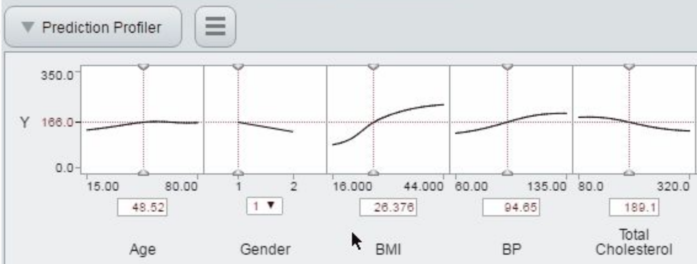
- PyTorch is one of our main libraries for generating geometries.
- Integrated a neural network as the data is queried to the webapp and handled by Jinja2.
- We can use the queried data to help with creating the images of the structure of the geometries.

Plotly.js



Plot Drop Volume, Drop Velocity, Refill, and Overshoot volume as a function of ink viscosity and an overlay of chamber thickness. Create a drop-down table to overlay multiple geometries (Architecture ID's)

Used Prediction Profiler to help create the plots



Y 166.0

Age 48.52

Gender 1

BMI 26.376

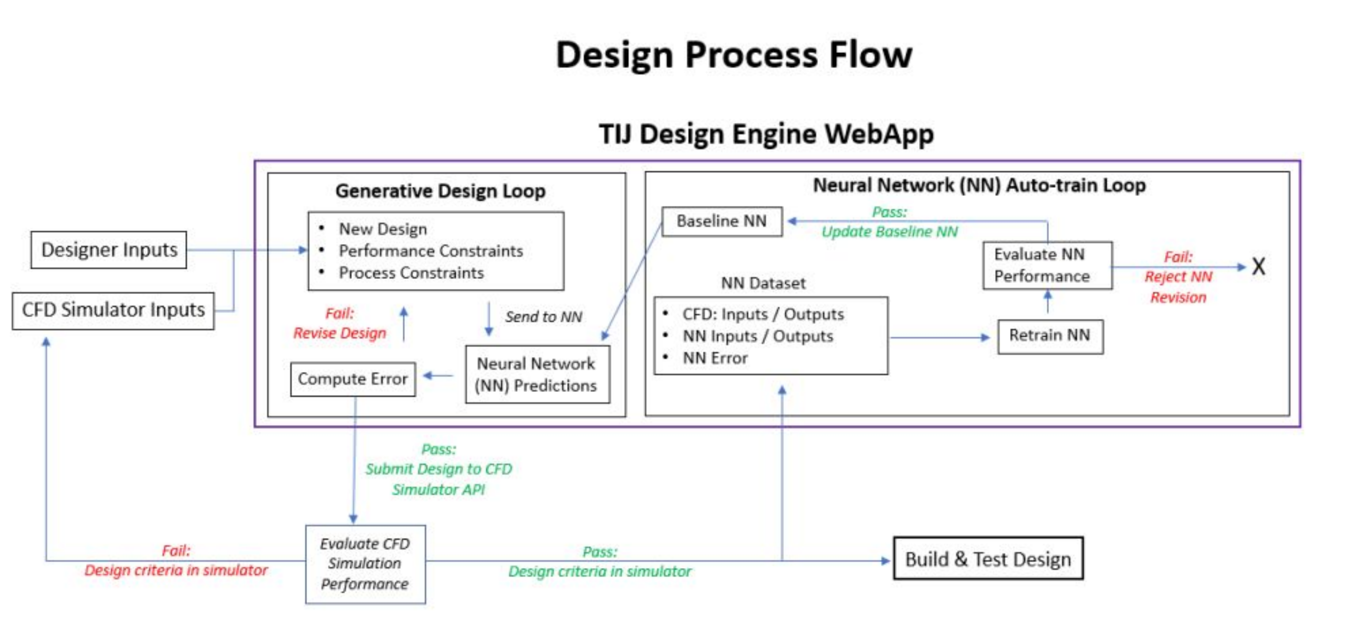
BP 94.65

Total Cholesterol 189.1

Challenges throughout development

- Working/learning with Flask
- Web development with Python
- Speed up NN with Generative design algorithms
    - Understanding generative design algorithms
    - Applying generative design to entire project flow
- Working with lack of guidance
    - Break tasks up into sizeable chunks
    - Distributing work into team
    - Understanding original code, learning flow of code
- Storing and passing values (HTML forms)
    - Passing the values into Flask
- Creating stock database model (SQL)
    - Working with querying the DB

Process Diagrams and Figures (1)



Design Process Flow

TIJ Design Engine WebApp

Designer Inputs

CFD Simulator Inputs

Generative Design Loop

- New Design
- Performance Constraints
- Process Constraints

Fail:
Revise Design

Send to NN

Compute Error

Neural Network (NN) Predictions

Neural Network (NN) Auto-train Loop

Baseline NN

Pass:
Update Baseline NN

Evaluate NN Performance

Fail:
Reject NN Revision

X

NN Dataset

- CFD: Inputs / Outputs
- NN Inputs / Outputs
- NN Error

Retrain NN

Pass:
Submit Design to CFD Simulator API

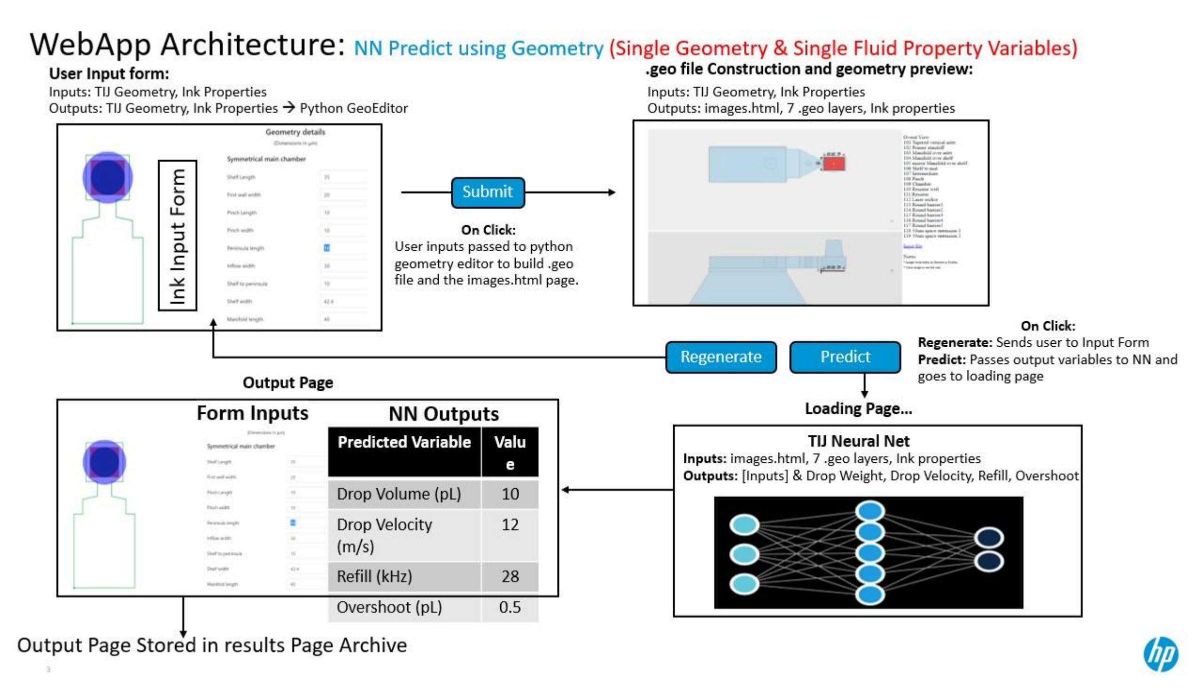
Fail:
Design criteria in simulator

Evaluate CFD Simulation Performance

Pass:
Design criteria in simulator

Build & Test Design

Process Diagrams and Figures (2)



WebApp Architecture: NN Predict using Geometry (Single Geometry & Single Fluid Property Variables)

User Input form:
Inputs: TIJ Geometry, Ink Properties
Outputs: TIJ Geometry, Ink Properties → Python GeoEditor

Geometry details
(Dimensions in µm)
Symmetrical main chamber

Shelf Length	15
First wall width	20
Pinch Length	10
Pinch width	10
Peninsula length	[illegible]
Inflow width	30
Shelf to peninsula	10
Shelf width	42.4
Manifold length	40

On Click:
User inputs passed to python geometry editor to build .geo file and the images.html page.

.geo file Construction and geometry preview:
Inputs: TIJ Geometry, Ink Properties
Outputs: images.html, 7 .geo layers, Ink properties

On Click:
Regenerate: Sends user to Input Form
Predict: Passes output variables to NN and goes to loading page

Loading Page...

TIJ Neural Net
Inputs: images.html, 7 .geo layers, Ink properties
Outputs: [Inputs] & Drop Weight, Drop Velocity, Refill, Overshoot

Output Page

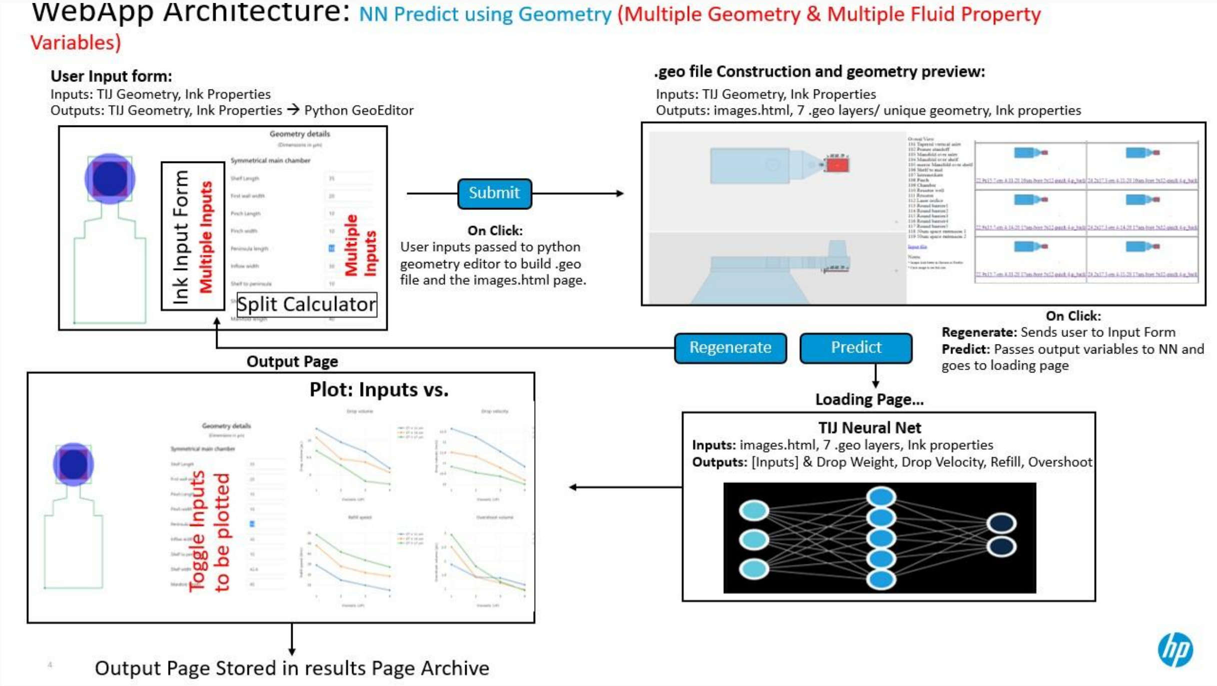
Form Inputs

NN Outputs

Predicted Variable	Value
Drop Volume (pL)	10
Drop Velocity (m/s)	12
Refill (kHz)	28
Overshoot (pL)	0.5

Output Page Stored in results Page Archive

Process Diagrams and Figures (3)



WebApp Architecture: NN Predict using Geometry (Multiple Geometry & Multiple Fluid Property Variables)

User Input form:
Inputs: TIJ Geometry, Ink Properties
Outputs: TIJ Geometry, Ink Properties → Python GeoEditor

.geo file Construction and geometry preview:
Inputs: TIJ Geometry, Ink Properties
Outputs: images.html, 7 .geo layers/ unique geometry, Ink properties

On Click:
User inputs passed to python geometry editor to build .geo file and the images.html page.

On Click:
Regenerate: Sends user to Input Form
Predict: Passes output variables to NN and goes to loading page

Loading Page…

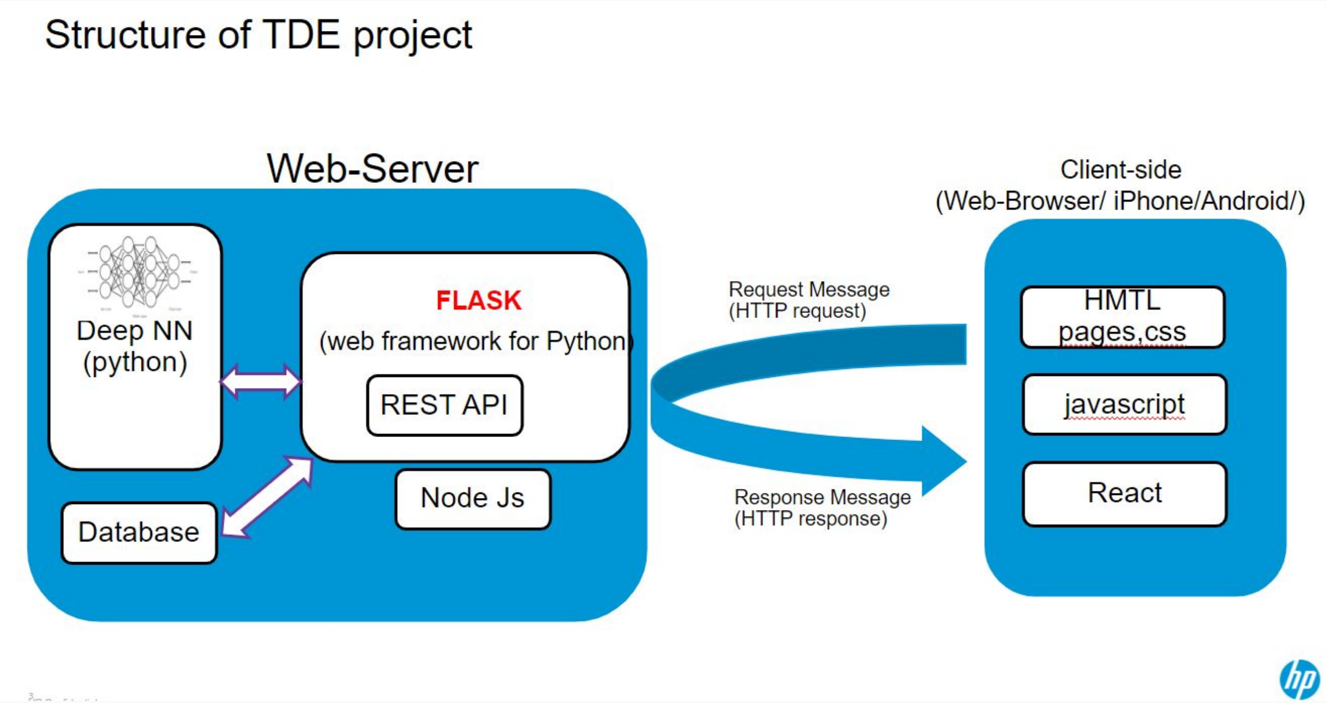
TIJ Neural Net
Inputs: images.html, 7 .geo layers, Ink properties
Outputs: [Inputs] & Drop Weight, Drop Velocity, Refill, Overshoot

Output Page

Plot: Inputs vs.

Output Page Stored in results Page Archive

Process Diagrams and Figures (4)



Structure of TDE project

Web-Server

Deep NN
(python)

FLASK
(web framework for Python)

REST API

Node Js

Database

Request Message
(HTTP request)

Response Message
(HTTP response)

Client-side
(Web-Browser/ iPhone/Android/)

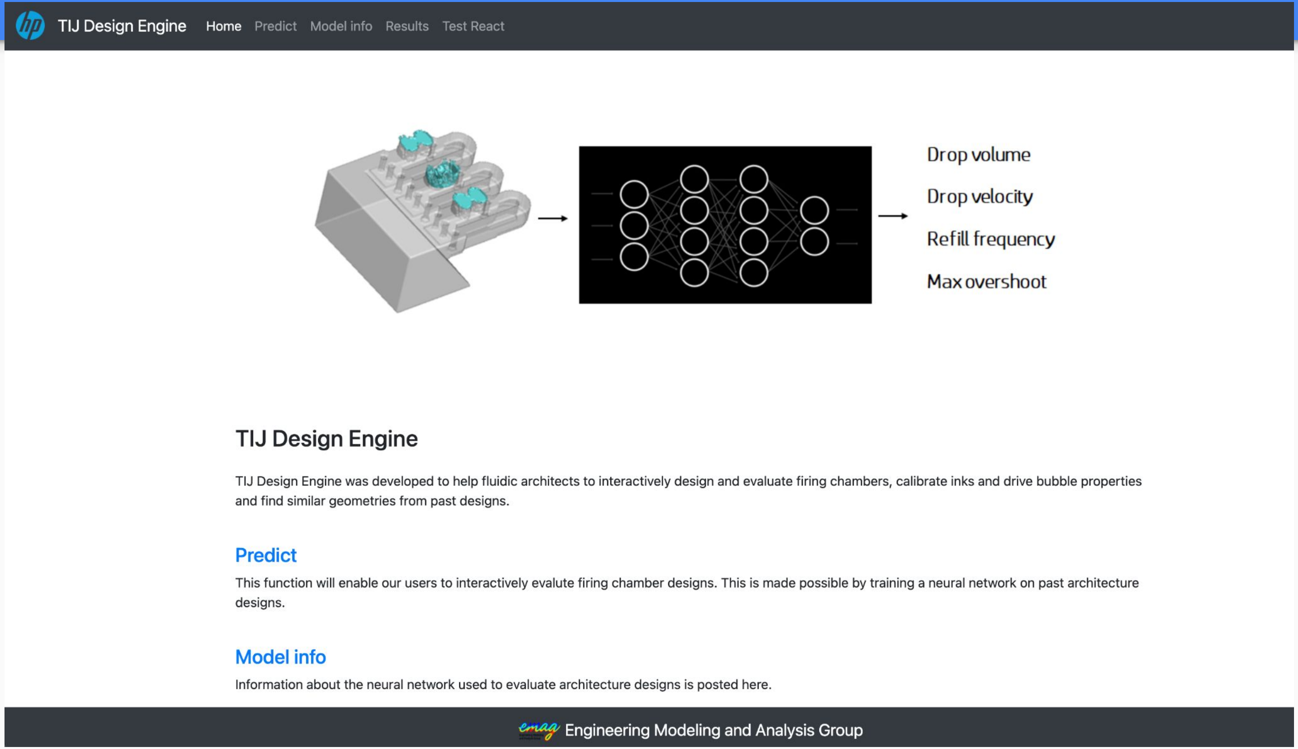
HMTL
pages,css

javascript

React

Demo (Webapp Screenshots)

- Home screen



TIJ Design Engine

TIJ Design Engine was developed to help fluidic architects to interactively design and evaluate firing chambers, calibrate inks and drive bubble properties and find similar geometries from past designs.

Predict

This function will enable our users to interactively evalute firing chamber designs. This is made possible by training a neural network on past architecture designs.

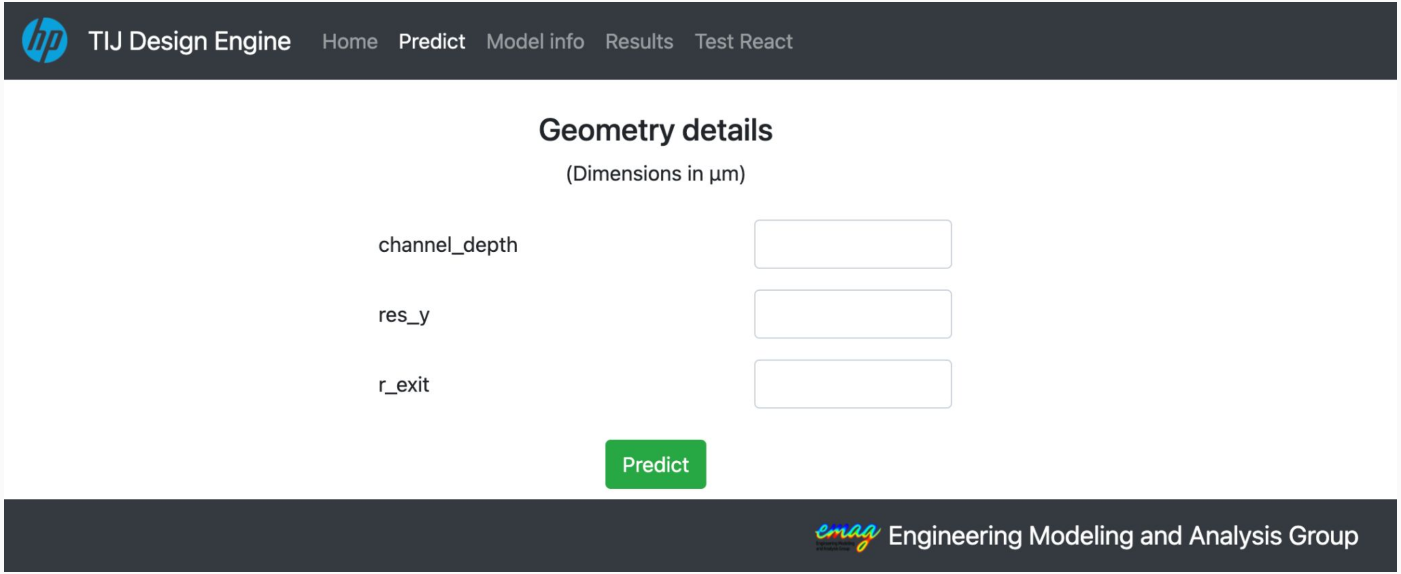
Model info

Information about the neural network used to evaluate architecture designs is posted here.

emag Engineering Modeling and Analysis Group

Predict Page

- Predict page: Form for our 3 fluid parameters being passed to cfd3 scripts



Geometry details

(Dimensions in µm)

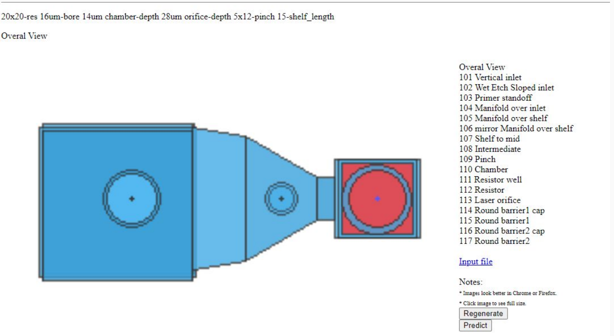
channel_depth

res_y

r_exit

Prediction Preview

- Preview of geometry



20x20-res 16um-bore 14um chamber-depth 28um orifice-depth 5x12-pinch 15-shelf_length

Overal View

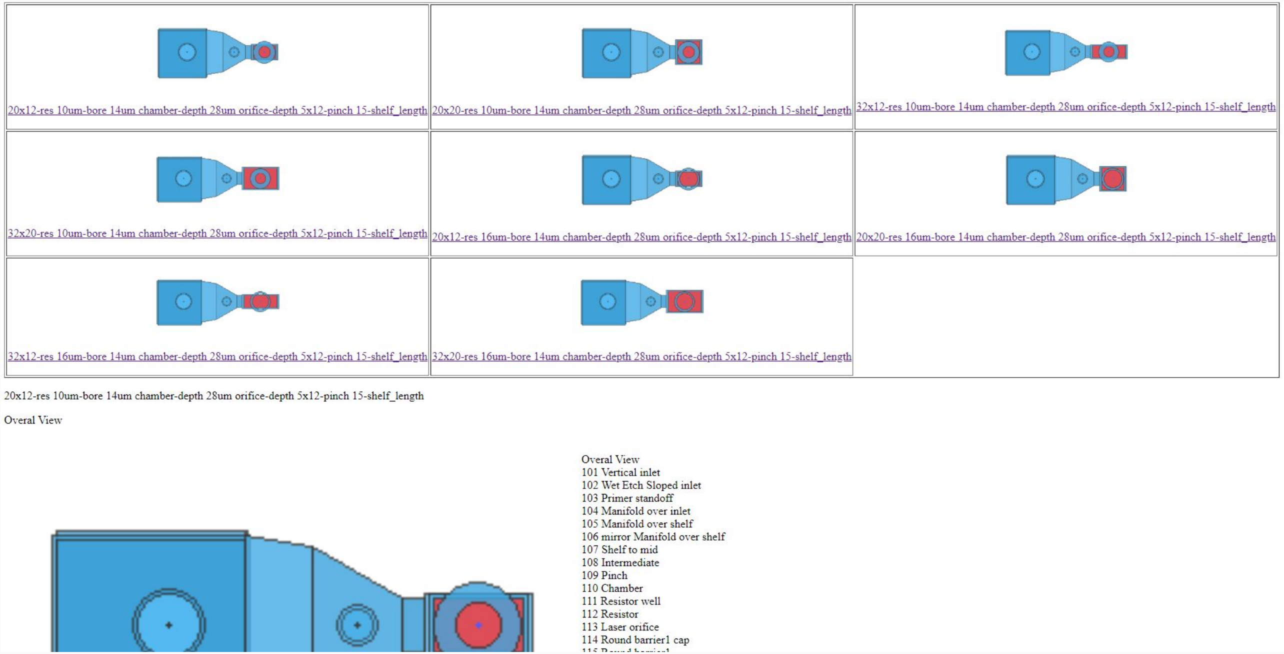
Overal View
101 Vertical inlet
102 Wet Etch Sloped inlet
103 Primer standoff
104 Manifold over inlet
105 Manifold over shelf
106 mirror Manifold over shelf
107 Shelf to mid
108 Intermediate
109 Pinch
110 Chamber
111 Resistor well
112 Resistor
113 Laser orifice
114 Round barrier1 cap
115 Round barrier1
116 Round barrier2 cap
117 Round barrier2

Input file

Notes:

* Images look better in Chrome or Firefox.

* Click image to see full size.



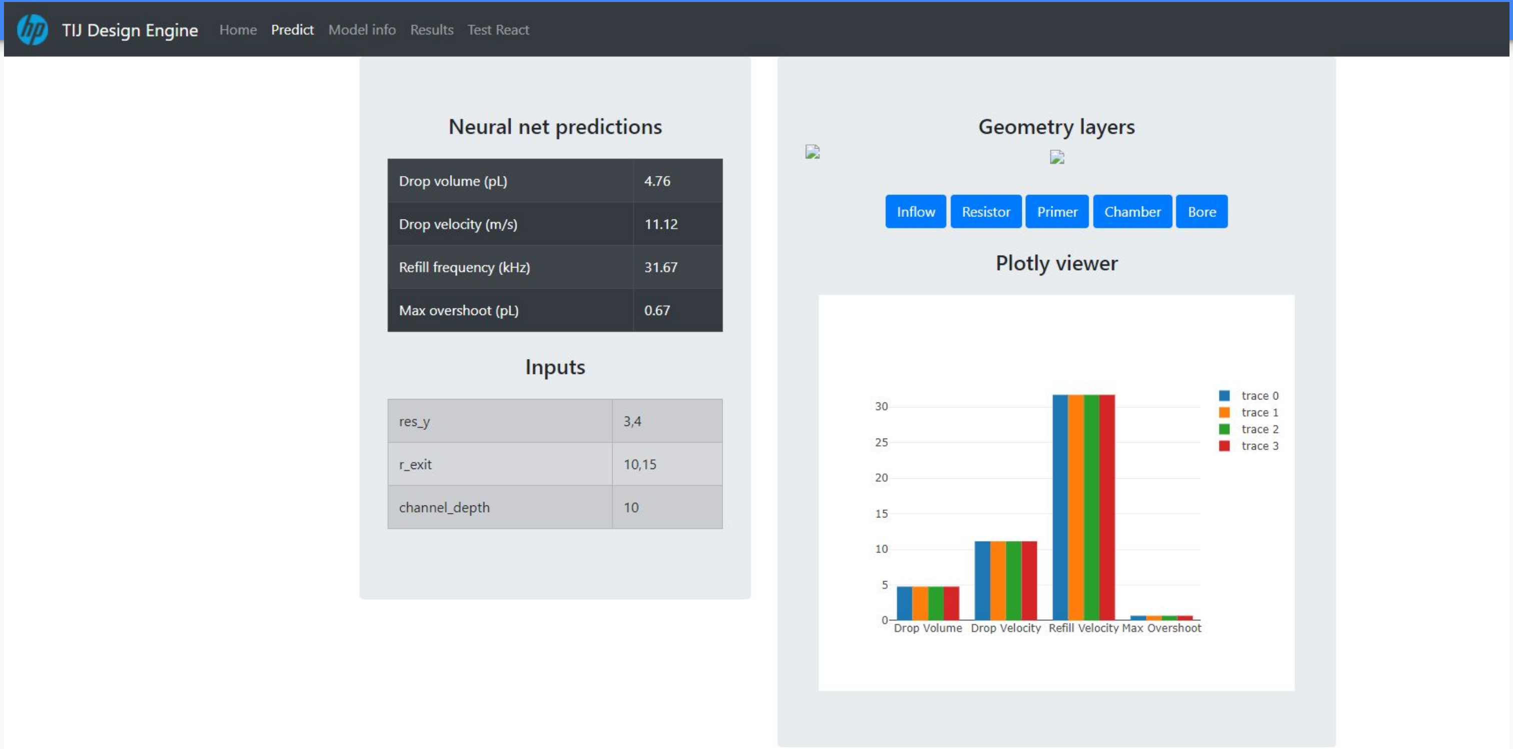
20x12-res 10um-bore 14um chamber-depth 28um orifice-depth 5x12-pinch 15-shelf_length	20x20-res 10um-bore 14um chamber-depth 28um orifice-depth 5x12-pinch 15-shelf_length	32x12-res 10um-bore 14um chamber-depth 28um orifice-depth 5x12-pinch 15-shelf_length
32x20-res 10um-bore 14um chamber-depth 28um orifice-depth 5x12-pinch 15-shelf_length	20x12-res 16um-bore 14um chamber-depth 28um orifice-depth 5x12-pinch 15-shelf_length	20x20-res 16um-bore 14um chamber-depth 28um orifice-depth 5x12-pinch 15-shelf_length
32x12-res 16um-bore 14um chamber-depth 28um orifice-depth 5x12-pinch 15-shelf_length	32x20-res 16um-bore 14um chamber-depth 28um orifice-depth 5x12-pinch 15-shelf_length	

20x12-res 10um-bore 14um chamber-depth 28um orifice-depth 5x12-pinch 15-shelf_length

Overal View

Overal View
101 Vertical inlet
102 Wet Etch Sloped inlet
103 Primer standoff
104 Manifold over inlet
105 Manifold over shelf
106 mirror Manifold over shelf
107 Shelf to mid
108 Intermediate
109 Pinch
110 Chamber
111 Resistor well
112 Resistor
113 Laser orifice
114 Round barrier1 cap

Output Page (NN query, Plotly.js)



Neural net predictions

Drop volume (pL)	4.76
Drop velocity (m/s)	11.12
Refill frequency (kHz)	31.67
Max overshoot (pL)	0.67

Inputs

res_y	3,4
r_exit	10,15
channel_depth	10

Geometry layers

Plotly viewer

Recent successes

- Isolating *cfd3* python script for single use
- *predict_input* values passed to *cfd3* python script
- Continue work on *predict_output*, querying NN for prediction values
- Continue templating codebase
- Update *predict_preview* with new, correct ‘images.html’
- *Predict_output* geometry viewer

User Stories

- “As a microfluidic designer I need to automate and simplify the process of data collection and documentation that is required to create designs more effectively and efficiently. I need tools that are scale-able and grow relevancy and capability with time, userbase, and data generation. In the long term I would like to automate most of the design and layout workflow.”
- We have successfully simplified the data collection process in order to create new geometries easily.
  - Input parameter support (res_y, r_exit, channel_depth) found in Predict page
- Webapp will maintain relevance with time and data generation
- Increased automation compared to start.

Thank you