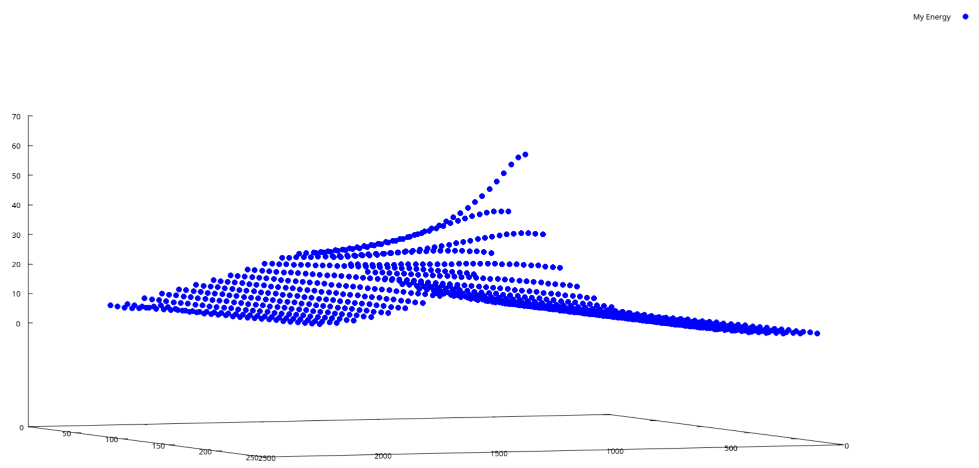
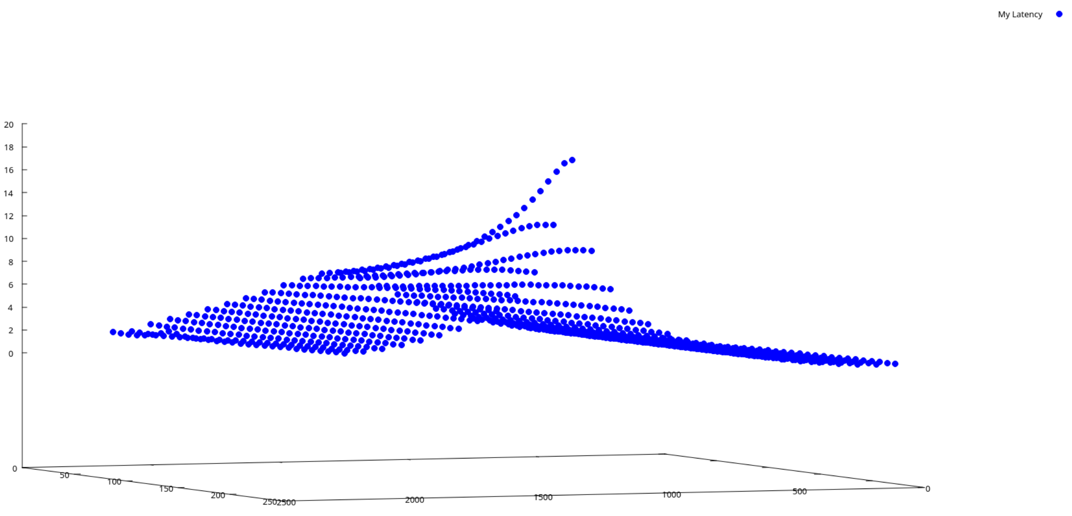
I chose to approximate the relationship between kernel configuration parameters (height×width dimension and input channels) and performance metrics (power, energy, and latency) for the add_relu operation across several model configurations.

Inputs: Height×Width (HW), Input Channels (CIN)

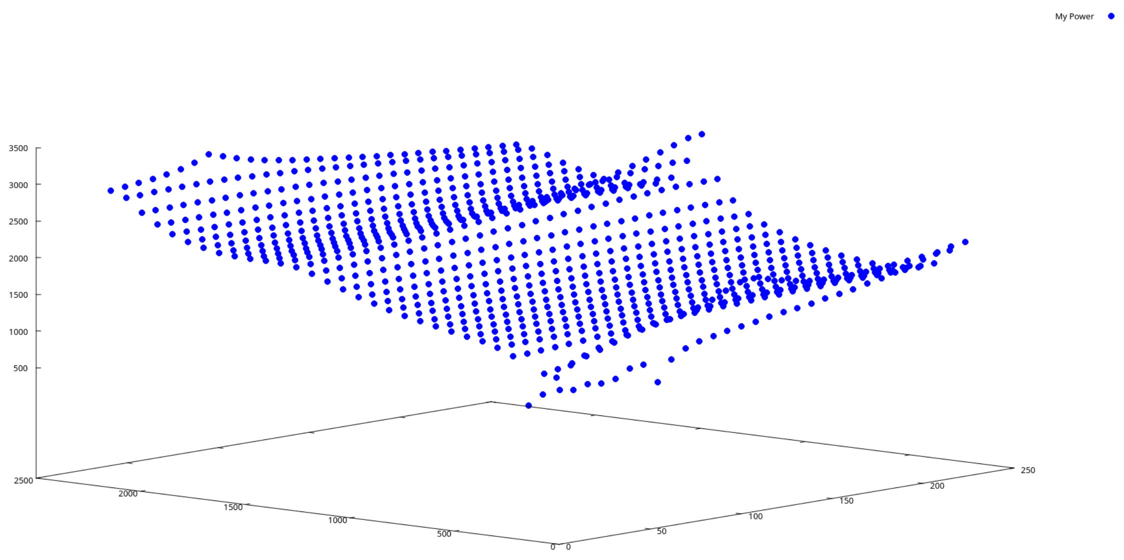
Outputs: Power (mW), Energy (μJ), Latency (ms)



Energy



Latency



Power

Based on initial data visualization, the 3D plots demonstrated a linear relationship with positive correlation between HW, CIN, and all three output metrics. There seemed to be no need to use a complicated model like MLP or CNN as there is no spatial relationship between the data. Linear regression is well-suited for this application because it is computationally efficient, requires little memory, and is easy to debug. Linear regression has minimal training overhead. The model trains via a closed-form solution (the normal equation: β = (X^T X)^(-1) X^T Y), requiring only matrix operations rather than iterative optimization. Training time is effectively instantaneous even on CPU. Once trained, the model stores only a small coefficient matrix (3×3 in this case) - approximately 9x8=72 bytes. Inference requires a single matrix multiplication, making it ideal for resource-constrained or real-time scenarios. There is no back-propagation involved. Linear regression finds the best-fit linear relationship between inputs and outputs by minimizing the sum of squared errors. For our multi-input, multi-output case, it learns three separate equations:

$\text{Power} = \beta_0^{(1)} + \beta_1^{(1)} \cdot \text{HW} + \beta_2^{(1)} \cdot \text{CIN}$

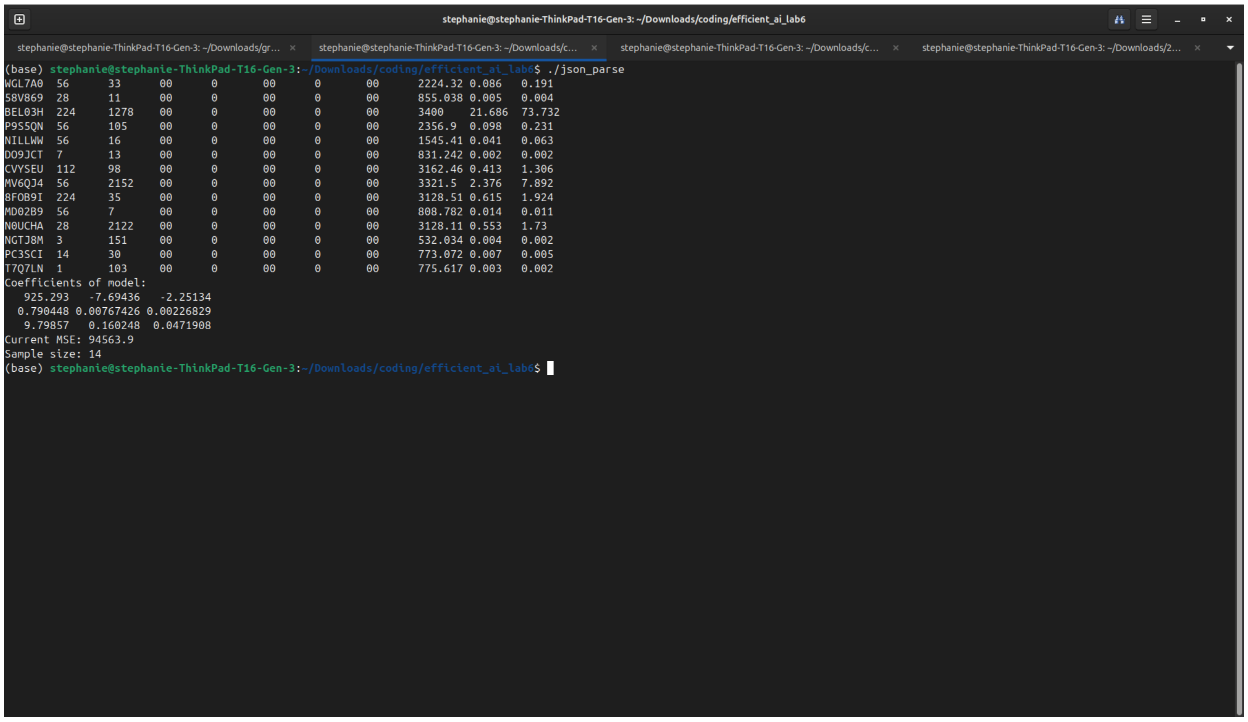
$\text{Energy} = \beta_0^{(2)} + \beta_1^{(2)} \cdot \text{HW} + \beta_2^{(2)} \cdot \text{CIN}$

$\text{Latency} = \beta_0^{(3)} + \beta_1^{(3)} \cdot \text{HW} + \beta_2^{(3)} \cdot \text{CIN}$

The algorithm constructs a design matrix X with our input features plus an intercept term, and solves for coefficients that best predict the output matrix Y. The normal equation guarantees finding the globally optimal solution in one step.

Current MSE: 94,563.9

This high error is primarily due to the mismatch in scale between power, latency, and energy. As you can see from the following output where the rightmost three columns are in order: power, latency, energy:



```
(base) stephanie@stephanie-ThinkPad-T16-Gen-3:~/Downloads/coding/efficient_ai_lab6$ ./json_parse
WGL7A0  56      33      00      0       00      0       00      2224.32 0.086   0.191
58V869  28      11      00      0       00      0       00      855.038 0.005   0.004
BEL03H  224     1278    00      0       00      0       00      3400    21.686  73.732
P9S5QN  56      105     00      0       00      0       00      2356.9  0.098   0.231
NILLWW  56      16      00      0       00      0       00      1545.41 0.041   0.063
DO9JCT  7       13      00      0       00      0       00      831.242 0.002   0.002
CVYSEU  112     98      00      0       00      0       00      3162.46 0.413   1.306
MV6QJ4  56      2152    00      0       00      0       00      3321.5  2.376   7.892
8FOB9I  224     35      00      0       00      0       00      3128.51 0.615   1.924
MD02B9  56      7       00      0       00      0       00      808.782 0.014   0.011
N0UCHA  28      2122    00      0       00      0       00      3128.11 0.553   1.73
NGTJ8M  3       151     00      0       00      0       00      532.034 0.004   0.002
PC3SCI  14      30      00      0       00      0       00      773.072 0.007   0.005
T7Q7LN  1       103     00      0       00      0       00      775.617 0.003   0.002
Coefficients of model:
   925.293   -7.69436   -2.25134
  0.790448 0.00767426 0.00226829
   9.79857   0.160248  0.0471908
Current MSE: 94563.9
Sample size: 14
(base) stephanie@stephanie-ThinkPad-T16-Gen-3:~/Downloads/coding/efficient_ai_lab6$
```

power is much larger than the other values.

For the following lab I will work on the following:

Feature normalization: Standardize inputs and outputs to comparable scales (z-score normalization or min-max scaling)

Proper train/test split: Reserve 20-30% of data for unbiased evaluation

Expanded dataset: Include more samples from add_relu and other operations

The linear model provides a strong baseline with negligible overhead, making it a suitable starting point for kernel performance prediction.