Building AI Models to Improve Medical Diagnosis

Emilio Daza

Dartmouth College

January 10, 2025

Motivation

Main Problem

"About 12 million people in the U.S. are misdiagnosed in outpatient care every year." - Harvard School of Public Health

AI as a promising solution to misdiagnoses

Convolutional Neural Network performed better at detecting melanomas in comparison to 58 dermatologists from 17 countries (The International Oncology Network, 2018)

Today's talk

- Heart Disease Diagnosis AI and Medical Imaging Interpreter
  - Mathematical Algorithms
  - Results
- Contribution
  - First model outperformed some other algorithms at detecting absence of heart disease
  - Second model achieved 94.55% accuracy in detecting meningioma tumors. The second best method got an accuracy of 85.64%.

First Model: Heart Disease Diagnosis

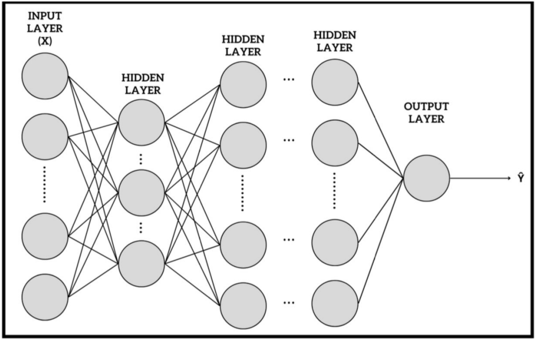


Figure 1: *Simplification of the Neural Network Architecture*

Initial Layers

Input Layer: Cardiovascular Risk Factors and Indicators

Columns: # of samples; Rows: # of features (symptoms)

$$X^{[0]} = \begin{pmatrix} x_{1,1} & \cdots & x_{1,237} \\ x_{2,1} & \cdots & x_{2,237} \\ \cdots & \ddots & \cdots \\ x_{13,1} & \cdots & x_{13,237} \end{pmatrix}$$

First Hidden Layer Calculation:

$$Z^{[1]} = W^{[1]}X^{[0]} + B^{[1]}$$

- $Z^{[n]}$: Matrix representation of the n layer
- $W^{[n]}$: Randomly initialized weights matrix
- $B^{[n]}$: Randomly initialized bias matrix

Sample Information to Matrix | Matrices Dimensionalities

Forward Propagation

1) $Z^{[1]} = W^{[1]}X^{[0]} + B^{[1]} :=$ Layer 1

2) $ReLU(x) = \begin{cases} x & \text{if } x > 0 \\ 0 & \text{if } x \leq 0 \end{cases}$

3) $A^{[1]} = ReLU(Z^{[1]})$

4) $Z^{[2]} = W^{[2]}A^{[1]} + B^{[2]} :=$ Layer 2

5) Process is repeated until reaching the output layer composed of one neuron

6) $Sigmoid(z) = \frac{1}{1+e^{-z}}, D : (-\infty, \infty), R : (0, 1)$

7) Binary Classifications: $\hat{Y} = \begin{cases} 1 & \text{if } Sigmoid(z) \geq 0.5 \\ 0 & \text{if } Sigmoid(z) < 0.5 \end{cases}$

Backpropagation

Loss Function: Binary Cross Entropy Loss

$$BCE = -\frac{1}{N}\sum_{i=0}^{N}[y_i log(\hat{y}_i) + (1 - y_i) log(1 - \hat{y}_i)]$$

- y_i: Actual target (0 or 1)
- $\hat{y}_i$: Predicted probability of the target (not yet rounded, that is just the $Sigmoid(z)$ of the last layer)
- N: # Samples fed to the model at a time

- We have to sufficiently minimize the loss function. Let's use partial derivates and the chain rule to do it.

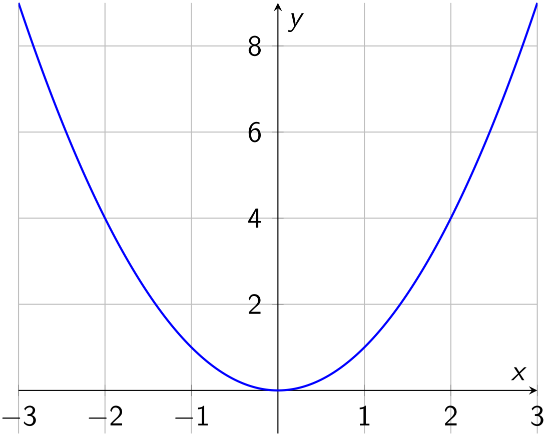
$$\frac{\partial BCE}{\partial W} = \frac{\partial BCE}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial Z} \cdot \frac{\partial Z}{\partial W}$$
$$\frac{\partial BCE}{\partial B} = \frac{\partial BCE}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial Z} \cdot \frac{\partial Z}{\partial B}$$

- We can update the weights' and bias' parameters to make the loss get closer to zero. (This happens in every layer from the end to the beginning)

$$\text{New } W^{[5]}_{numbers} = W^{[5]}_{numbers} - \alpha \frac{\partial BCE}{\partial W^{[5]}_{variables}}(W^{[5]}_{numbers})$$
$$\text{New } B^{[5]}_{numbers} = B^{[5]}_{numbers} - \alpha \frac{\partial BCE}{\partial B^{[5]}_{variables}}(B^{[5]}_{numbers})$$

Note: One random sample is loaded at a time (SGD).
Definition: An epoch is when all the samples in the training dataset have gone through the model.

Gradient Descent Intuition



Determination of Number of Epochs

Accuracy Analysis

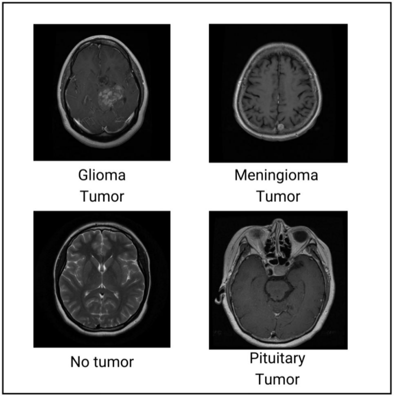
$$VD_{acc,n} = \frac{PTE_n}{TT_{VD}} \cdot 100$$
$$95\%\ \text{CI Margin}_n = \left(196\sqrt{\frac{VD_{acc,n}(1-VD_{acc,n})}{N}}\right)$$

- PTE_n = Correct Predicted Targets after the Epoch n
- $VD_{acc,n}$ = Validation Data Accuracy
- TT_{VD} = Total Targets in the Validation Dataset (same as # samples in VD)

Note: The data was standarized but regarding their corresponding feature and the dataset was split in three: X_{test}, X_{train}, and X_{val}

Second Model: Medical Imaging Interpreter

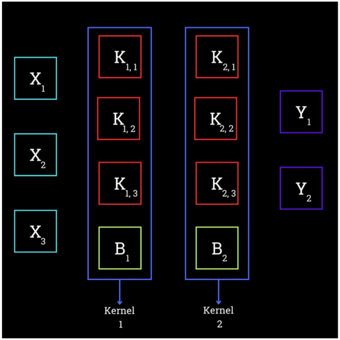
Four classes:



Convolutional Neural Network (CNN)

Convolutional Layer

Example:



Valid Cross-Correlation

$$Y_1 = B_1 + X_1 \star K_{1,2} + X_2 \star K_{1,2} + X_3 \star K_{1,3}$$

$$Y_2 = B_2 + X_1 \star K_{2,1} + X_2 \star K_{2,2} + X_3 \star K_{2,3}$$

In general,

$$Y_1 = B_1 + X_1 \star K_{1,2} + X_2 \star K_{1,2} + \cdots + X_n \star K_{1,n}$$

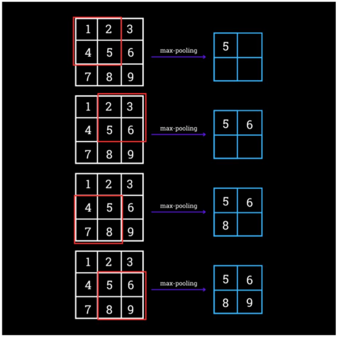
$$Y_2 = B_2 + X_1 \star K_{2,1} + X_2 \star K_{2,2} + \cdots + X_n \star K_{1,n}$$

$$\cdots$$

$$Y_d = B_d + X_1 \star K_{d,1} + X_2 \star K_{d,2} + \cdots + X_n \star K_{d,n}$$

Other types of layers used

Max-pooling:



Average-pooling: Same process as max-pooling but computing the average

Backpropagation

One Hot Encoding:

- glioma_tumor: [1,0,0,0]
- meningioma_tumor: [0,1,0,0]
- no_tumor: [0,0,1,0]
- pituitary_tumor: [0,0,0,1]

Cross Entropy Loss:

$$CEL = -\left(\frac{1}{n}\right) \sum_{i=1}^{n} \sum_{j=1}^{c} (y\_true_{i,j}) log(y\_pred_{i,j})$$

- n = Number of samples in the batch
- i = index of the sample (ranges from 1 to 16)
- j = index of possible labels (ranges from 1 to 4)
- $y\_true_{i,j}$ = true label for sample i and label j
- $y\_pred_{i,j}$ = predicted probability for sample i and label j

The parameters' updates of the outer layer would look like this (slightly modified because of momentum):

$$K_{i,j}{}^{(3)}_{numbers} \leftarrow K_{i,j}{}^{(3)}_{numbers} - \alpha \frac{\partial L}{\partial K_{i,j}{}^{(3)}_{variables}} (K_{i,j}{}^{(3)}_{numbers})$$
$$B_{i,j}{}^{(3)}_{numbers} \leftarrow B_{i,j}{}^{(3)}_{numbers} - \alpha \frac{\partial L}{\partial B_{i,j}{}^{(3)}_{variables}} (B_{i,j}{}^{(3)}_{numbers})$$

In order to change the parameters for hidden layers, keep in mind that $Y^{(2)} = X^{(3)}$.

$$\frac{\partial L}{\partial K_{i,j}^{(2)}} = X_j^{(2)} \star \frac{\partial L}{\partial Y_i^{(2)}} = X_j^{(2)} \star \frac{\partial L}{\partial X_i^{(3)}}$$
$$\frac{\partial L}{\partial B_i^{(2)}} = \frac{\partial L}{\partial Y_i^{(2)}} = \frac{\partial L}{\partial X_j^{(3)}}$$

The same algorithm can be used to find the partials of the other hidden layers and update those parameters.

What's momentum?

$$V_t = \beta V_{t-1} + \alpha \nabla_{w_t} L(W_t, X, y)\ ,\ \ W_{t+1} = W_t - V_t$$

- $\nabla_{w_t} L(W_t, X, y)$:Gradient of the Loss Function w.r.t a learnable parameter (will be applied to all contained in the model)
- α: Learning rate
- V_t: Velocity at time step t
- W_t: A model parameter at time step t
- β: Momentum coefficient

Note: V_{t-1} is initialized as $V_0 = 0$. Accelerates convergence.

Both models' specific details

Note:

$$ReLU(x) = \begin{cases} 0 & \text{if } x \leq 0 \\ x & \text{if } x > 0 \end{cases}, \quad ReLU'(x) = \begin{cases} 0 & \text{if } x \leq 0 \\ 1 & \text{if } x > 0 \end{cases}$$

<u>Workflows</u>:

First Model:

- Layers' output neurons:
  - First (after the input layer): 360
  - Second: 180
  - Third: 90
  - Fourth: 45
  - Fifth (last): 1
- Activation functions: ReLU, Sigmoid (end)
- Learning Rate: 0.01
- Loss: BCE
- Optimization Algorithm: SGD
- # Epochs: 482

Second Model:

1. Conv. Layer: 16 kernels with K matrices of shape 3x3; ReLU; Batch Normalization, Max-pooling
2. The same process is repeated but now the inputs are the results obtained.
3. After that, the same happens with the additional step of average pooling.
4. Then, all the information is converted into a matrix where each row is a sample, and each column the information extracted per sample.
5. Standard Feedforward Neural Network: first layer with 120 neurons, ReLU, Second Layer with 84 neurons, ReLu, output layer with 4 neurons. Output Layer: 16x4 matrix where each row is a sample and each column the predicted probability for the classes.

Second model details

- Learning rate: 0.01
- Momentum coefficient: 0.9
- Batch Normalization: Each batch of size 16 is fed to the network

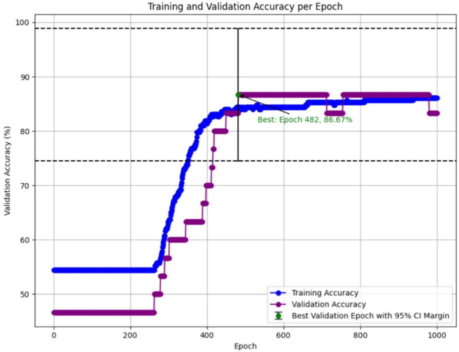
$$\hat{x} = \frac{x - \bar{x}}{\sqrt{V(x) + 10^{-5}}}, \quad y = \gamma \hat{x} + \beta$$

where γ and β are learnable parameters

Results

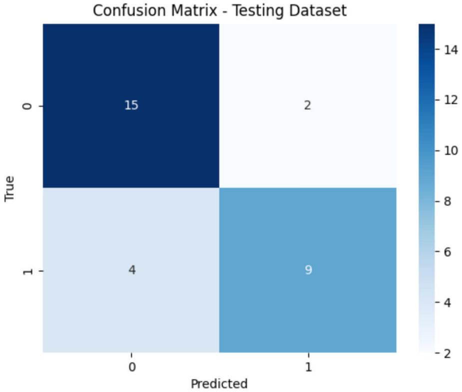
First Model: Heart Disease Diagnosis AI

Epoch selection



Data Sizes:
Training: 237 (79.8%)
Validation: 30 (10.1%)
Testing: 30 (10.1%)
Validation Accuracy: 86.67%

Confusion Matrix for First Model (Testing Dataset)



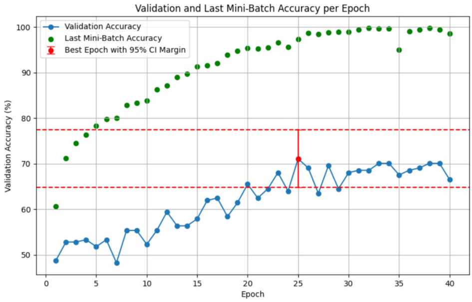
Note: 0 indicates absence and 1 presence.

- Overall Accuracy: 80%
- Absence: 88.2%
- Presence: 69.2%

Results

Second Model: Medical Imaging Interpreter AI

Epoch selection



Data Sizes:
Training: 2870 (87.92%)
Validation: 197 (6.03%)
Testing: 197 (6.03%)
Validation Accuracy: 70.56%

Confusion Matrix for Second Model (Testing Dataset)

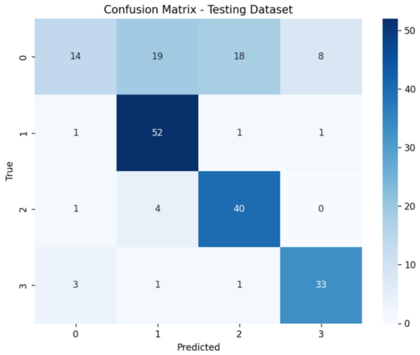


Figure: 0 represents Glioma Tumor, 1 Meningioma Tumor, 2 No Tumor, and 3 Pituitary Tumor

- Overall Accuracy: 71.07%
- Meningioma: 94.55%
- No Tumor: 88.89%
- Pituitary: 86.8%
- Glioma: 23.7%

Discussion and Conclusion

- **First Model:** The meaningful contribution of the model is that it outperforms other proposed machine learning algorithms published from the scientific community at detecting absence (Nashif, Raihan, Islam, & Imam, 2018) such as an artificial neural network by 24.78% and a Naive Bayes algorithm by 5.32%.
- **Second Model:** For the four-class classification task, the model achieves 94.55% accuracy in detecting meningioma tumors. As a reference, the second-best method (Google Vision Transformer) from recent machine learning research projects in the community acquired a 85.64% accuracy for the same task.

Special Thanks

This material is based upon work supported by the National Science Foundation under Grant No. 2015553.
Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.
Research Mentors: Peter Mucha and Rebecca Hardenbrook

References I

3Blue1Brown. (2017). *But what is a neural network? — Chapter 1, Deep Learning.* [Online]. Available: https://www.youtube.com/watch?v=aircAruvnKk. Last Time Accessed: August 28th, 2024.

Adam Conner-Simons. (2020). *An automated health care system that understands when to step in.* Harvard-MIT Health Sciences and Technology. [Online]. Available: https://hst.mit.edu/news-events/automated-health-care-system-understands-when-step. Last Time Accessed: August 28th, 2024.

References II

- Aditya Mittal. (2011). *Lanczos Resampling Digital Signal Image Processing - Aditya Mittal - AcmeTutor - Sinc function.* [Online]. Available: `https://www.youtube.com/watch?v=62nPlSZszOg&t=105s`. Last Time Accessed: August 29th, 2024.

- Andras Janosi, William Steinbrunn, Matthias Pfisterer, and Robert Detrano. (1988). *Heart Disease UC Irvine Machine Learning Repository.* [Online]. Available: `https://archive.ics.uci.edu/dataset/45/heart+disease`. Last Time Accessed: August 28th, 2024.

References III

Animated AI. (2022). *Kernel Size and Why Everyone Loves 3x3 - Neural Network Convolution.* [Online]. Available: `https://www.youtube.com/watch?v=V9ZYDCnItr0`. Last Time Accessed: August 29th, 2024.

Coding Lane. (2021). *Forward Propagation and Backward Propagation — Neural Networks — How to train Neural Networks.* [Online]. Available: `https://www.youtube.com/watch?v=Tb23YtZ92AE&t=161s`. Last Time Accessed: August 28th, 2024.

References IV

DeepLearningAI. (2017). *Understanding Mini-Batch Gradient Dexcent (C2W2L02).* [Online]. Available: https://www.youtube.com/watch?v=-_4Zi8fCZO4&t=399s. Last Time Accessed: August 29th, 2024.

Emma Roth. (2023). *Microsoft spent hundreds of millions of dollars on a ChatGPT supercomputer.* The Verge. [Online]. Available: https://www.theverge.com/2023/3/13/23637675/microsoft-chatgpt-bing-millions-dollars-supercomputer-c
Last Time Accessed: August 29th, 2024.

References V

Futurology - An Optimistic Future. (2020). *Convolutional Neural Networks Explained (CNN Visualized).* [Online]. Available: https://www.youtube.com/watch?v=pj9-rr1wDhM. Last Time Accessed: August 29th, 2024.

Gianluca Turcatel. (2021). *Derivation of the Binary Cross Entropy Loss Gradient.* Python Unleashed. [Online]. Available: https://www.python-unleashed.com/post/derivation-of-the-binary-cross-entropy-loss-gradient. Last Time Accessed: August 29th, 2024.

References VI

Greg Hogg. (2023). *Deep Learning Hyperparameter Tuning in PyTorch — Making the Best Possible ML Model — Tutorial 2.* [Online]. Available: `https://www.youtube.com/watch?v=xP9l9MptIZo`. Last Time Accessed: August 29th, 2024.

Karen Feldscher. (2019). *The doctors will see you now.* Harvard School of Public Health. [Online]. Available: `https://www.hsph.harvard.edu/news/features/improving-doctors-diagnostic-accuracy/`. Last Time Accessed: August 28th, 2024.

References VII

Milind Sahay. (2020). *Neural Networks and the Universal Approximation Theorem.* [Online]. Available: https://towardsdatascience.com/neural-networks-and-the-universal-approximation-theorem

Last Time Accessed: August 29th, 2024.

mtpc4s9. (2024). *Brain Tumor Prediction - Google Vision Transformer.* [Online]. Available: https://www.kaggle.com/code/mtpc4s9/brain-tumor-prediction-google-vision-transformer.

Last accessed: December 17, 2024.

References VIII

Patrick Loeber. (2021). *Deep Learning With PyTorch - Full Course.* [Online]. Available: https://www.youtube.com/watch?v=c36lUUr864M&t=6296s. Last Time Accessed: August 29th, 2024.

Peter Hofland. (2018). *Artificial Intelligence Better than Dermatologists in Diagnosing Skin Cancer.* The International Oncology Network. [Online]. Available: https://www.oncozine.com/artificial-intelligence-better-dermatologists-diagnosin
Last Time Accessed: August 28th, 2024.

References IX

- PyTorch Contributors. (2023). *BatchNorm2d.* PyTorch. [Online]. Available: `https://pytorch.org/docs/stable/generated/torch.nn.BatchNorm2d.html`. Last Time Accessed: August 29th, 2024.
- PyTorch Contributors. (2023). *MaxPool2d.* PyTorch. [Online]. Available: `https://pytorch.org/docs/stable/generated/torch.nn.MaxPool2d.html`. Last Time Accessed: August 29th, 2024.
- PyTorch Contributors. (2023). *AvgPool2d.* PyTorch. [Online]. Available: `https://pytorch.org/docs/stable/generated/torch.nn.AvgPool2d.html`. Last Time Accessed: August 29th, 2024.

References X

Rina. (2020). *Overfitting in Machine Learning.* MTI Technology. [Online]. Available: https://ailab.mti-vietnam.vn/blog/2020/12/04/overfitting-in-machine-learning/. Last Time Accessed: August 30th, 2024.

Samson Zhang. (2020). *Building a neural network FROM SCRATCH (no Tensorflow/Pytorch, just numpy & math).* [Online]. Available: https://www.youtube.com/watch?v=w8yWXqWQYmU&t=120s. Last Time Accessed: August 28th, 2024.

References XI

Sarah Moore. (2023). *Lab Automation in Clinical Diagnostics.* AZO Life Sciences. [Online]. Available: https://www.azolifesciences.com/article/Lab-Automation-in-Clinical-Diagnostics.aspx. Last Time Accessed: August 28th, 2024.

Sartaj Bhuvaji, Ankita Kadam, Prajakta Bhumkar, Sameer Dedge, and Swati Kanchan. (2020). *Brain Tumor Classification (MRI).* Kaggle. DOI: 10.34740/KAGGLE/DSV/1183165 [Online]. Available: https://www.kaggle.com/dsv/1183165.

References XII

Shadman Nashif, Md. Rakib Raihan, Md. Rasedul Islam, and Mohammad Hasan Imam. (2018). *Heart Disease Detection by Using Machine Learning Algorithms and a Real-Time Cardiovascular Health Monitoring System.* World Journal of Engineering and Technology, 6(4), 854–873. doi:10.4236/wjet.2018.64057 [Online]. Available: `https://www.scirp.org/journal/paperinformation?paperid=88650`. Last accessed: December 17, 2024.

StatQuest with Josh Starmer. (2019). *Gradient Descent, Step-by-Step.* [Online]. Available: `https://www.youtube.com/watch?v=sDv4f4s2SB8`. Last Time Accessed: August 29th, 2024.

References XIII

The Independent Code. (2021). *Convolutional Neural Network from Scratch — Mathematics & Python Code.* [Online]. Available: https://www.youtube.com/watch?v=Lakz2MoHy6o. Last Time Accessed: August 29th, 2024.

Vitaly Bushaev. (2017). *Stochastic Gradient Descent with momentum.* Medium. [Online]. Available: https://towardsdatascience.com/stochastic-gradient-descent-with-momentum-a84097641a5d. Last Time Accessed: August 29th, 2024.

Appendix Slides - Matrix presentation of sample information

Consider 237 samples as vectors (there are 13 features per each):

$$\begin{aligned}
s_1 &= < v_{1,1}, v_{1,2}, \ldots, v_{1,13} > \\
s_2 &= < v_{2,1}, v_{2,2}, \ldots, v_{2,13} > \\
&\ldots \\
s_{237} &= < v_{237,1}, v_{237,2}, \ldots, v_{237,13} >
\end{aligned}$$

Each node in the input layer represents the data per feature:

$$\begin{aligned}
x_1 &= < v_{1,1}, v_{2,1}, \ldots, v_{237,1} > \\
x_2 &= < v_{1,2}, v_{2,2}, \ldots, v_{237,2} > \\
&\ldots \\
x_{13} &= < v_{1,13}, v_{2,13}, \ldots, v_{237,13} >
\end{aligned}$$

Appendix Slides - Matrix dimensionalities

Note:

- # Rows in weights', bias matrix, and z matrix: # Neurons in the layer
- # Cols in weights' matrix: # Neurons in the previous layer.
- # Cols in the z matrix and 1's matrix: # Samples in the previous layer

$$\begin{bmatrix} z_{1,1} & \cdots & z_{1,237} \\ z_{2,1} & \cdots & z_{2,237} \\ \vdots & \ddots & \vdots \\ z_{360,1} & \cdots & z_{360,237} \end{bmatrix} = \begin{bmatrix} w_{1,1} & \cdots & w_{1,13} \\ w_{2,1} & \cdots & w_{2,13} \\ \vdots & \ddots & \vdots \\ w_{360,1} & \cdots & w_{360,13} \end{bmatrix} \begin{bmatrix} x_{1,1} & \cdots & x_{1,237} \\ x_{2,1} & \cdots & x_{2,237} \\ \vdots & \ddots & \vdots \\ x_{13,1} & \cdots & x_{13,237} \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_{360} \end{bmatrix} \begin{bmatrix} 1_1 & 1_2 & \dots & 1_{237} \end{bmatrix}$$

Further Reading

- **Complete Research Paper:** Daza Vigo, E. S. (2023). *Machine Learning Approaches for Precision Medicine*. Retrieved from `https://www.researchgate.net/publication/383692584_Machine_Learning_Approaches_for_Precision_Medicine`

You can also scan the QR code to access the paper.