Graduation Project Report

Spring 2024

Hanheng He (400485161)

Course Instructor: Dr. Antoine Deza

April 10, 2024

Contents

1 Introduction

Let matrix $A \in \mathbb{R}^{n\times n}$ is invertible, with the *spectral norm* defined as $||A||_s = sup_{x\neq 0}|Ax|/|x|$, the *condition number* of A is $c(A) = ||A||_s||A^{-1}||_s$. The *condition number* is a measurement of the sensibility of the equation $Ax = b$ when right hand side is changed. If $c(A)$ is large, then A is called $ill-conditioned$[[1](???)].

In [[2](???)], researcher restricted entries into set $\{0, 1\}$ or $\{-1, 1\}$, denoted by $\mathcal{A}_n^1$ or $\mathcal{A}_n^2$. With these conditions, many quantities are equivalent to the condition number. Let A be a non-singular $(0, 1)$ matrix, $B = A^{-1} = (b_{ij})$, the following quantity as an equivalent condition number is considered in [[2](???)]:

$$\chi(A) = max_{i,j}|b_{ij}|.$$

In this report, we aim to construct a ill-conditioned $(0, 1)$ matrix C satisfied [[3](???)]

$$\chi(C) \geq 2^{\frac{1}{2}n\log n - n(2+o(1))},$$

that is said, matrix C has a controlled lower bound with respect to n. We started from generating matrix $A \in \mathcal{A}_n^2$, and we generate $B \in \mathcal{A}_n^1$ based on A. With a series of matrix B with different shapes, we can further more concatenate a matrix C with it's condition number controlled.

2 Generate Ill-conditioned Matrix C

2.1 Generate Set Ω

In order to generate matrix $A \in \mathcal{A}_n^2$, we need to firstly generate set Ω. Let $|\cdot|$ denotes cardinality and Δ denote symmetric different. Let $m \in \mathbb{Z}^+$, $n = 2^m$, $\Omega = \{\alpha_0, \alpha_1, \alpha_2, ..., \alpha_n\}$ be a set of $n + 1$ element such that $|\alpha_i| \leq |\alpha_{i+1}|$ and $|\alpha_i\Delta\alpha_{i+1}| \leq 2$. Let $\alpha_0 = \{\varnothing\}$. We also have $\alpha_1 = \{\varnothing\}$. Suppose we have $\Omega = \{\{\varnothing\}, \{\varnothing\}, \{1\}, \{2\}, ..., \{m\}\}$ initially. Considering for every time we only take out all the sets with maximum size in Ω, and insert only one element inside by order. That is, for an existed set $\{1\}$, we insert $2, 3, ..., m$ by order.

With this way of insertion, we only need to consider if the conditions are met between the last old set and the first new set, and the continuous new sets come from different old sets. For all the sets come from the same old set, the conditions are automatically met.

```
Initially Omega = {{}, {}, {1}, {2}, ..., {m}}, take {1}, {2}, ..., {m} out,
and insert only one element. Make sure conditions are met between
    1. {m} and {1, _};
    2. {1, _} and {2, _}, {2, _} and {3, _}, ...
```

For the first case above, the only element we can insert is m, which is the first element if we revert the ordered insertion. For the second case, let's say we have set α with size k and β with size $k - 1$, if $\alpha \cup \beta = \alpha$, we can insert any element we wish; if $\alpha \cup \beta \neq \alpha$, we can only insert an element $r \in \alpha$. Since we always insert element in order or in reversed order, if the first element of the insertion list is not in α, the first element of the reverted insertion list must be in α *(WHY???)*.

The following pseudo code shows the way generating Ω:

```
let Omega = {{}, {}, {1}, {2}, ..., {m}};
for i in {1, 2, ..., m - 1}:
    let sets = sets in Omega with size equals i
    for set in sets:
        let avail_range = [max(set) + 1, ..., m]
        if avail_range[0] not in Omega[-1]:
            avail_range = avail_range.reverse()
        for element in avail_range:
            Omega.append({set..., element}) // set... means extend the set
```

when $m = 4$, set Ω is shown below:

```
Omega = {set(), set(), {1}, {2}, {3}, {4},
        {1, 4}, {1, 3}, {1, 2}, {2, 4}, {2, 3},
        {3, 4}, {1, 3, 4}, {1, 2, 3}, {1, 2, 4},
        {2, 3, 4}, {1, 2, 3, 4}}
```

2.2 Generate Matrix $A \in \mathcal{A}_n^2$

Shown in [[1](???)], with set Ω satisfying given conditions, we can generate matrix $A \in \mathcal{A}_n^2$ such that $\chi(A) = 2^{\frac{1}{2}n\log n - n(1+o(1))}$. Let matrix $A \in \mathcal{A}_n^2$, with set $\Omega = \{\alpha_0, \alpha_1, \alpha_2, ..., \alpha_n\}$ satisfies $|\alpha_i| \leq |\alpha_{i+1}|$ and $|\alpha_i \Delta \alpha_{i+1}| \leq 2$, matrix A can be constructed as follows[[1](???)]. For every $1 \leq i, j \leq n$:

$$a_{ij} = \begin{cases} -1, \; \alpha_j \bigcap (\alpha_{i-1} \bigcup \alpha_i) = \alpha_{i-1}\Delta\alpha_i \; and \; |\alpha_{i-1}\Delta\alpha_i| = 2 \\ (-1)^{|\alpha_{i-1} \bigcap \alpha_j| + 1}, \; \alpha_j \bigcap (\alpha_{i-1} \bigcup \alpha_i) \neq \varnothing \; but \; does \; not \; meet \; the \; condition \; above \\ 1, \; \alpha_j \bigcap (\alpha_{i-1} \bigcup \alpha_i) = \varnothing \end{cases}.$$

With the Ω shown in section 1.1 *(should be a link here)*, the matrix $A \in \mathcal{A}_n^2$ constructed is shown below:

$$A = \begin{bmatrix}
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & -1 & 1 & 1 & 1 & -1 & -1 & -1 & 1 & 1 & 1 & -1 & -1 & -1 & 1 & -1 \\
1 & 1 & -1 & 1 & 1 & 1 & 1 & -1 & -1 & -1 & 1 & 1 & -1 & -1 & -1 & -1 \\
1 & 1 & 1 & -1 & 1 & 1 & -1 & 1 & 1 & -1 & -1 & -1 & -1 & 1 & -1 & -1 \\
1 & 1 & 1 & 1 & -1 & -1 & 1 & 1 & -1 & 1 & -1 & -1 & 1 & -1 & -1 & -1 \\
1 & -1 & 1 & 1 & 1 & 1 & -1 & -1 & 1 & 1 & 1 & 1 & -1 & 1 & 1 & 1 \\
1 & 1 & 1 & -1 & 1 & -1 & 1 & 1 & 1 & -1 & -1 & -1 & 1 & -1 & -1 & -1 \\
1 & 1 & -1 & 1 & 1 & 1 & -1 & 1 & -1 & -1 & 1 & -1 & -1 & 1 & -1 & -1 \\
1 & 1 & 1 & 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 & -1 & -1 & -1 & 1 & -1 \\
1 & 1 & 1 & -1 & 1 & 1 & -1 & 1 & -1 & 1 & -1 & -1 & 1 & -1 & -1 & -1 \\
1 & 1 & 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 & -1 & -1 \\
1 & -1 & 1 & 1 & 1 & 1 & 1 & -1 & 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\
1 & 1 & -1 & 1 & 1 & -1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 & -1 & -1 & 1 \\
1 & 1 & 1 & 1 & -1 & 1 & -1 & -1 & 1 & -1 & -1 & -1 & 1 & -1 & -1 & 1 \\
1 & 1 & 1 & -1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 & -1 & 1 \\
1 & -1 & 1 & 1 & 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 & -1 & -1 & 1 & 1
\end{bmatrix}$$

With the construction step above, it can be proved that $A = LQ$, where Q is a n by n matrix given by $q_{ij} = (-1)^{|\alpha_i \bigcap \alpha_j|}$, and Q is a symmetric Hadamard matrix, that is $Q^2 = QQ^T = nI_n$; $L = AQ^{-1}$ is a lower triangular matrix.

With matrix A shown above, matrix Q and L is shown below:

$$Q = \begin{bmatrix}
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & -1 & 1 & 1 & 1 & -1 & -1 & -1 & 1 & 1 & 1 & -1 & -1 & -1 & 1 & -1 \\
1 & 1 & -1 & 1 & 1 & 1 & 1 & -1 & -1 & -1 & 1 & 1 & -1 & -1 & -1 & -1 \\
1 & 1 & 1 & -1 & 1 & 1 & -1 & 1 & 1 & -1 & -1 & -1 & -1 & 1 & -1 & -1 \\
1 & 1 & 1 & 1 & -1 & -1 & 1 & 1 & -1 & 1 & -1 & -1 & 1 & -1 & -1 & -1 \\
1 & -1 & 1 & 1 & -1 & 1 & -1 & -1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 \\
1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\
1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 & 1 \\
1 & 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 & 1 \\
1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 & 1 \\
1 & 1 & 1 & -1 & -1 & -1 & -1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 & 1 \\
1 & -1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 & -1 & 1 & -1 & 1 & 1 & 1 & -1 \\
1 & -1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 & 1 & -1 \\
1 & -1 & -1 & 1 & -1 & 1 & -1 & 1 & 1 & -1 & -1 & 1 & 1 & -1 & 1 & -1 \\
1 & 1 & -1 & -1 & -1 & -1 & -1 & -1 & 1 & 1 & 1 & 1 & 1 & 1 & -1 & -1 \\
1 & -1 & -1 & -1 & -1 & 1 & 1 & 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 & 1
\end{bmatrix}$$

$$Q \times Q = \begin{bmatrix}
16 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 16 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 16 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 16 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 16 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 16 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 16 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 16 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 16 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 16 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 16 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 16 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 16 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 16 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 16 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 16
\end{bmatrix}$$

$$L = \begin{bmatrix}
1.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\
0.0 & 1.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\
0.0 & 0.0 & 1.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\
0.0 & 0.0 & 0.0 & 1.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\
0.0 & 0.0 & 0.0 & 0.0 & 1.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\
0.5 & 0.5 & 0.0 & 0.0 & -0.5 & 0.5 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\
0.0 & 0.0 & 0.0 & 0.5 & 0.5 & -0.5 & 0.5 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\
0.0 & 0.0 & 0.5 & 0.5 & 0.0 & 0.0 & -0.5 & 0.5 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\
0.0 & 0.5 & 0.0 & 0.0 & 0.5 & 0.0 & 0.0 & -0.5 & 0.5 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\
0.0 & 0.0 & 0.0 & 0.5 & 0.5 & 0.0 & 0.0 & 0.0 & -0.5 & 0.5 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\
0.0 & 0.0 & 0.5 & 0.0 & 0.5 & 0.0 & 0.0 & 0.0 & 0.0 & -0.5 & 0.5 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\
0.25 & 0.25 & 0.0 & 0.25 & 0.25 & 0.25 & 0.25 & 0.0 & 0.0 & 0.0 & -0.75 & 0.25 & 0.0 & 0.0 & 0.0 & 0.0 \\
0.0 & 0.0 & 0.25 & 0.0 & 0.25 & 0.25 & 0.0 & 0.25 & 0.0 & 0.25 & 0.25 & -0.75 & 0.25 & 0.0 & 0.0 & 0.0 \\
0.0 & 0.0 & 0.0 & 0.25 & 0.25 & 0.25 & 0.25 & 0.0 & 0.25 & 0.25 & 0.0 & 0.0 & -0.75 & 0.25 & 0.0 & 0.0 \\
0.0 & 0.25 & 0.0 & 0.25 & 0.0 & 0.25 & 0.0 & 0.25 & 0.0 & 0.25 & 0.25 & 0.0 & 0.0 & -0.75 & 0.25 & 0.0 \\
0.125 & 0.125 & 0.125 & 0.125 & 0.125 & 0.125 & 0.125 & 0.125 & 0.125 & 0.125 & 0.125 & 0.125 & 0.125 & 0.125 & -0.875 & 0.125
\end{bmatrix}$$

2.3 Generate Matrix $B \in \mathcal{A}^1_{n-1}$

With matrix $A \in \mathcal{A}^2_n$ be constructed, a mapping can be implemented to generate a matrix $A \in \mathcal{A}^1_{n-1}$[[1](???)]. Consider the map Φ which assigns to any matrix $B \in \mathcal{A}^1_{n-1}$ a matrix $\Phi(B) \in \mathcal{A}^2_n$ in the following way:

$$\Phi(B) = \begin{pmatrix} 1 & 1_{n-1} \\ -1^T_{n-1} & 2B - J_{n-1} \end{pmatrix}.$$

It is obvious $\Phi(B)$ is a mapping $\mathcal{A}^2_n \to \mathcal{A}^1_{n-1}$ with a series of linear operations. Therefore, we have the following reversing way to construct matrix $B \in \mathcal{A}^1_{n-1}$ with $A = \{a_{ij}\} \in \mathcal{A}^2_n$:

$$B = \frac{1}{2}(J_{n-1} + \{a_{ij}\}_{2 \le i \le n, 2 \le j \le n}).$$

Notice that in the above section, the $A \in \mathcal{A}^2_n$ we constructed has it's first column as:

$$\begin{pmatrix} 1 \\ 1^T_{n-1} \end{pmatrix},$$

so we need to negative the first column, so finally we have the relation between matrix $B \in \mathcal{A}_{n-1}^1$ and $A \in \mathcal{A}_n^2$ in the implementation shown as follows:

$$B = \frac{1}{2}(J_{n-1} - \{a_{ij}\}_{2 \le i \le n, 2 \le j \le n}).$$

With the matrix $A \in \mathcal{A}_n^2$ constructed above, a constructed matrix B is shown below:

$$\begin{bmatrix}
1&0&0&0&1&1&1&0&0&0&1&1&1&0&1\\
0&1&0&0&0&0&1&1&1&0&0&1&1&1&1\\
0&0&1&0&0&1&0&0&1&1&1&1&0&1&1\\
0&0&0&1&1&0&0&1&0&1&1&0&1&1&1\\
1&0&0&0&0&1&1&0&0&0&0&1&0&0&0\\
0&0&1&0&1&0&0&0&1&1&1&0&1&1&1\\
0&1&0&0&0&1&0&1&1&0&1&1&0&1&1\\
0&0&0&1&1&0&1&0&0&1&1&1&1&0&1\\
0&0&1&0&0&1&0&1&0&1&1&0&1&1&1\\
0&0&0&1&1&0&0&1&1&0&0&1&1&1&1\\
1&0&0&0&0&0&1&0&0&1&1&0&0&1&1\\
0&1&0&0&1&1&0&1&0&1&0&1&1&1&0\\
0&0&0&1&0&1&1&0&1&1&1&0&1&1&0\\
0&0&1&0&1&1&1&1&0&0&1&1&0&1&0\\
1&0&0&0&0&0&0&1&1&1&1&1&1&0&0
\end{bmatrix}$$

Shown in [[1](???)], matrix $B \in \mathcal{A}_n^2$ preserves the property $\chi(B) = 2^{\frac{1}{2}n\log n - n(1+o(1))}$.

2.4 Generate and Verify Ill-conditioned Matrix C

With the sections above, it's sufficient to generate an ill-conditioned matrix C. Let S and T be two non-singular matrices of order n_1 and n_2. Define $S \diamond T$ as follows:

$$R = \begin{bmatrix}
s_{11} & \dots & s_{1n_1} & 0 & \dots & 0\\
s_{21} & \dots & s_{2n_1} & 0 & \dots & 0\\
\vdots & \vdots & \vdots & \vdots & \vdots & \\
s_{n_1 1} & \dots & s_{n_1 n_1} & 0 & \dots & 0\\
0 & 0 \dots 0 & 1 & t_{11} & \dots & t_{1n_2}\\
0 & 0 \dots 0 & 0 & t_{21} & \dots & t_{2n_2}\\
\vdots & \vdots & \vdots & \vdots & \vdots & \\
0 & 0 \dots 0 & 0 & t_{n_2 1} & \dots & t_{n_2 n_2}
\end{bmatrix}$$

it's shown in [[1](???)] that R has the following property:

$$\chi(S \diamond T) \ge \chi(S)\chi(S)$$

Now consider the $(0,1)$ matrix $C = A_1 \diamond (A_2 \diamond (\dots(A_{r-1} \diamond A_r))\dots)$, with[[1](???)]

$$\chi(C) \ge \prod_{i=1}^{r} \chi(A_i) > 2^{\frac{1}{2}n\log n - n(2+o(1))}$$

By showing the lower bound of $\chi(C)$, we can see matrix C is ill-conditioned with respect with n. The following block shows some result of $\chi(C)$ with C generate in the same method above related to order r.

```
r = 2, order of C = 4, χ(C) = 1.0
r = 3, order of C = 11, χ(C) = 2.0
r = 4, order of C = 26, χ(C) = 260.0
r = 5, order of C = 57, χ(C) = 106491641548.6
```

We can see that as order r grows, $\chi(C)$ grows rapidly.

3 Complexity Analysis of Generating Matrix C

To generate set Ω, everytime we only need to take out an generated set and insert one new element inside. Therefore, without cosidering the complexity of set insertion, the ideally time complexity is $O(n)$.

To generate every entry of matrix $A \in \mathcal{A}_n^2$, a visit of three continuous element in set Ω is necessary. Let n be the order of matrix A, without considering the complexity of set accessing, the total ideally time complexity is $O(n) + O(n^2) = O(n^2)$.

Generating matrix B is a simple matrix operation, let $n-1$ be the order of B, the time complexity is $O((n-1)^2) + O(n^2) = O(n^2)$.

In order to generate matrix C, we need a series of matrix A_1, A_2, ... , A_r. As shown above, for matrix A_r with order r, the time complexity is $O(r^2)$. Therefore the time complexity generating these matrices is $\sum_{i=1}^{n} O(r^2) = O(n^3)$

Therefore, when r grows, with an $O(n^3)$ time complexity, the time required to generate matrix C grows rapidly.

4 Conclusion

5 References

[1] Hashmi, M. F., Katiyar, S., Keskar, A. G., Bokde, N. D., & Geem, Z. W. (2020). Efficient pneumonia detection in chest xray images using deep transfer learning. *Diagnostics*, 10(6), 417.

[2] Palaz, D., & Collobert, R. (2015). *Analysis of CNN-based speech recognition system using raw speech as input* (No. REP_WORK). Idiap.

[3] Wang, J., Yu, L. C., Lai, K. R., & Zhang, X. (2016, August). Dimensional sentiment analysis using a regional CNN-LSTM model. In *Proceedings of the 54th annual meeting of the association for computational linguistics (volume 2: Short papers)* (pp. 225-230).

[4] He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 770-778).

[5]Tan, M., & Le, Q. (2019, May). Efficientnet: Rethinking model scaling for convolutional neural networks. In *International conference on machine learning* (pp. 6105-6114). PMLR.

[6] Li, L., Tan, Z., & Han, X. (2022). An Improved EfficientNet Model and its Applications in Pneumonia Image Classification. *Journal of Engineering Science & Technology Review*, 15(6).

[7] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., ... & Rabinovich, A. (2015). Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 1-9).

[8] Zhang, G., Hu, M. Y., Patuwo, B. E., & Indro, D. C. (1999). Artificial neural networks in bankruptcy prediction: General framework and cross-validation analysis. *European journal of operational research*, 116(1), 16-32.

[9] Kandel, I., & Castelli, M. (2020). The effect of batch size on the generalizability of the convolutional neural networks on a histopathology dataset. *ICT express*, 6(4), 312-315.