Investigating neural scaling laws in a multilayer perceptron

Leonid Elkin

```
        self.timer.start(16)

    def update_simulation(self):
        if not self.bodies: return
        n = len(self.bodies)

        total_mass = sum(b.mass for b in self.bodies)
        com_x = sum(b.x * b.mass for b in self.bodies) / total_mass
        com_y = sum(b.y * b.mass for b in self.bodies) / total_mass
        #Math part for all particles
        forces = [(0.0, 0.0)] * n
        for i in range(n):
            for j in range(n):
                if i == j: continue
                dx = self.bodies[j].x - self.bodies[i].x
                dy = self.bodies[j].y - self.bodies[i].y
                dist_sq = dx**2 + dy**2 + SOFTENING**2
                f = G * self.bodies[i].mass * self.bodies[j].mass / dist_sq
                dist = math.sqrt(dist_sq)
                fx, fy = f * dx / dist, f * dy / dist
                fx_total, fy_total = forces[i]
                forces[i] = (fx_total + fx, fy_total + fy)

        for i, body in enumerate(self.bodies):
            fx, fy = forces[i]
            body.vx += (fx / body.mass) * DT
            body.vy += (fy / body.mass) * DT
            body.x += body.vx * DT
            body.y += body.vy * DT

        #Dumb poopy colision detection
        survivors = []
        items = []
        for i, b in enumerate(self.bodies):
            if any(b != self.bodies[j] and b.distance_to(self.bodies[j].x, self.bodies[j].y) < COLLISION_RADIUS for j in range(len(self.bodies))):
                self.scene.removeItem(self.body_items[i])
                continue
            if self.destruction_checkbox.isChecked() and b.distance_to(com_x, com_y) > DESTRUCTION_RADIUS:
                self.scene.removeItem(self.body_items[i])
                continue
            survivors.append(b)
            items.append(self.body_items[i])
        self.bodies = survivors
        self.body_items = items

        #Mass counter
        mass_inside = sum(b.mass for b in self.bodies if b.distance_to(com_x, com_y) <= MASS_RADIUS)
        self.mass_display.setText(f"Mass within radius: {mass_inside:.2f}")
```

Investigating neural scaling laws in a multilayer perceptron

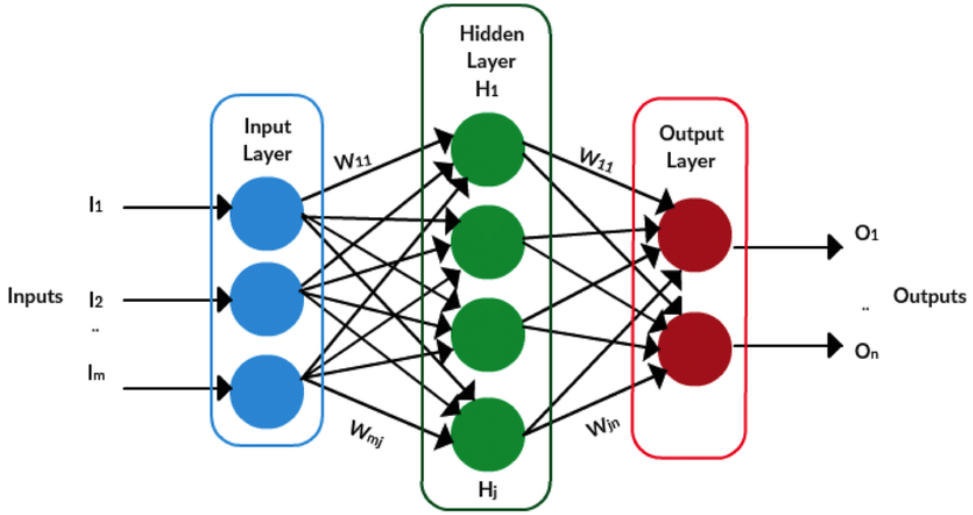
Context

- Multilayer perceptron – specific deep learning model.
- During this project I 'investigated' the network to achieve an optimal configuration. The results of that investigation are way too complicated to explain in this presentation so unfortunately, they won't be shown.
- This is an ARTEFACT although investigation-like features are present.

```
    beginning training with  {numEpochs} epochs and layer sizes {self.layerSizes}")

    epoch in range(numEpochs):
    lossEpochList = [] # shuffles training images
    shuffledIndexes = np.random.permutation(indexes)
    Inputs = inputs[shuffledIndexes]
    Labels = targets[shuffledIndexes]

    if epochLosses: # option to record losses for each epoch
        testPredictions = self.__forward(testInputs)
        testLoss = self.__loss(testPredictions, testLabels)
        epochLossesList.append(testLoss)
        print(testLoss)

    for batch in range(0, indexes, batchSize): # sorts out batch system
        batchInputs = Inputs[batch: (batch + batchSize)]
        batchLabels = Labels[batch: (batch + batchSize)]

        predictions = self.__forward(batchInputs, recordUsage) # sets up backpropagation
        loss = self.__loss(predictions, batchLabels)
        lossEpochList.append(loss)

        self.__backward(batchLabels, lr, recordUsage)


    lr *= decayRate

    avgLoss = np.mean(lossEpochList) # gets average loss for the epoch


    if recordUsage: # records CPU usage every epoch if boolean variable recordUsage is True
        testPredictions = self.__forward(testInputs)
        avgLoss = self.__loss(testPredictions, testLabels)
        usageSuperlist.append([avgLoss, self.CPUUsage])



    lossList.append(avgLoss) # appends average loss to a list

    if stopper: # if average loss stays the same MLP stops training early
        if len(lossList) >= 4:
            if ((lossList[-2] + lossList[-1]) / 2)  / ((lossList[-3] + lossList[-4]) / 2) == 1:
                print("Stopped")
                break

    print(f'Epoch {epoch + 1} / {numEpochs}, Average Loss: {avgLoss:.10f},CPU FLOPs: {self.CPUUsage} lr: {lr:.10f}')

# various things that I might want the MLP to return based on the input variables
f recordUsage:
    return usageSuperlist
```

What is my project

My project is on neural networks. The multilayer perceptron (or MLP) if one of the fundamental models that can be used for image recognition. As part of a research project to investigate various aspect of said model, I coded an implementation of it and compared the accuracy to others.

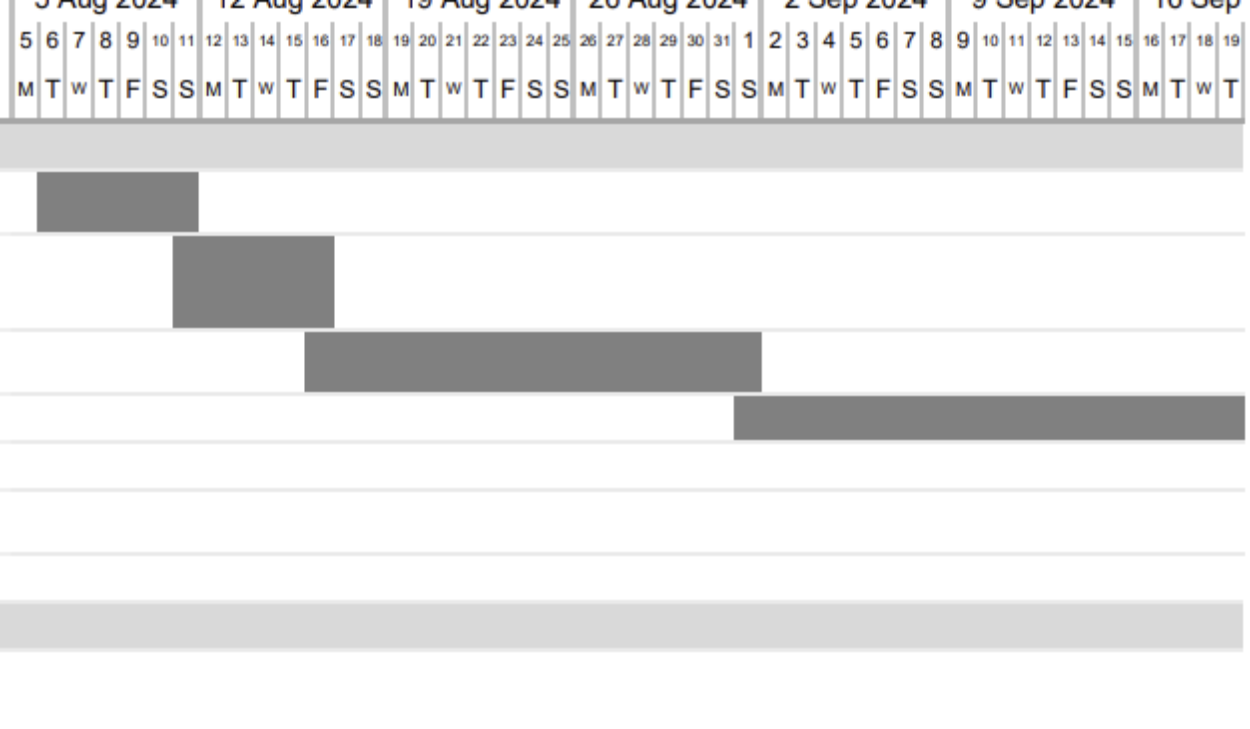


What inspired me to do this

- Interested in computer science – planning to do it or related subjects in university.
- Inspired to do a deep learning-based project after reading one of my brother's deep learning papers (Predicting public sentiment towards the Ukraine war).
- Initially I wanted to compare performances of two different models (MLP vs CNN).
- Changed project to be more like an investigation. The logic was to 'perfect one rather to do both'. Also the CNN is just an MLP with an extra layer of complexity.
- Final title is "Investigating neural scaling laws in a multilayer perceptron" - open ended project the difficulty of which I can adapt along the way.

Leonid Elkin

WBS	TASK	LEAD	START	END	DAYS	% DONE	WORK DAYS
1	**Preliminary research and planning**			-			-
1.1	Look at basic AI principles		Tue 8/06/24	Sun 8/11/24	6	100%	4
1.2	Research and start learning related libraries		Sun 8/11/24	Fri 8/16/24	5	100%	5
1.3	Reading up on more advanced principles		Fri 8/16/24	Sun 9/01/24	9	100%	7
1.4	Initial Ideas		Sun 9/01/24	Sun 9/22/24	15	100%	15
1.4.1	More research		Sun 9/22/24	Tue 10/01/24	7	100%	7
1.4.2	Candidate proposal A		Tue 10/01/24	Wed 10/16/24	12	100%	12
1.5	Planning review		Wed 10/16/24	Sat 11/23/24	28	100%	28
2	**Artefact creation and information gathering**			-			-
2.1	Getting training and testing data through primary research and		Fri 11/01/24	Sat 11/30/24	21	50%	21

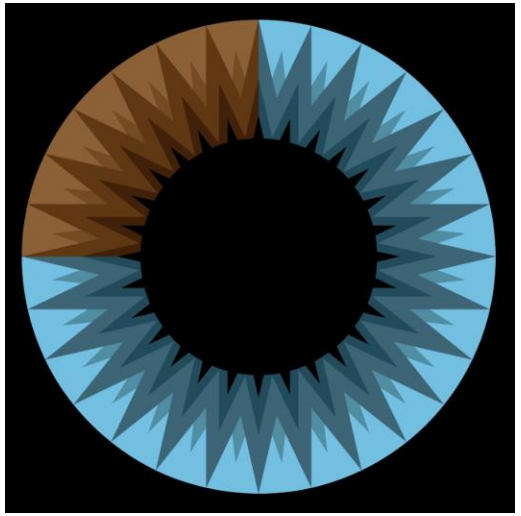


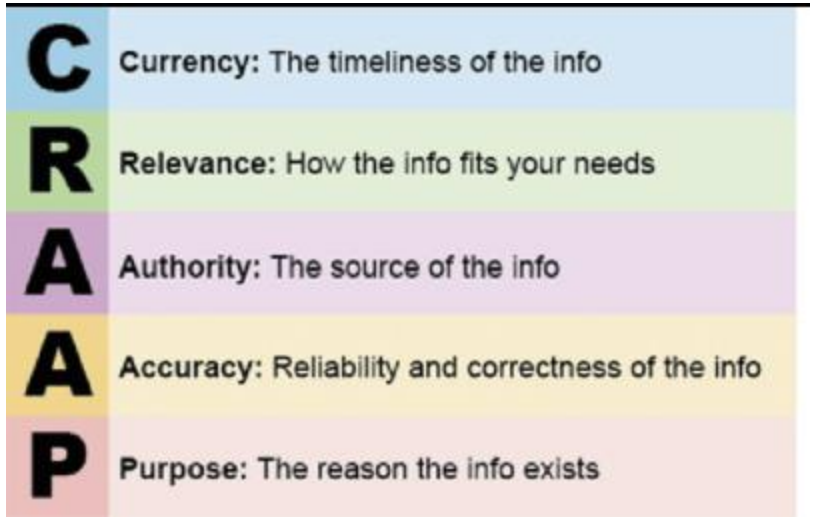
Planning

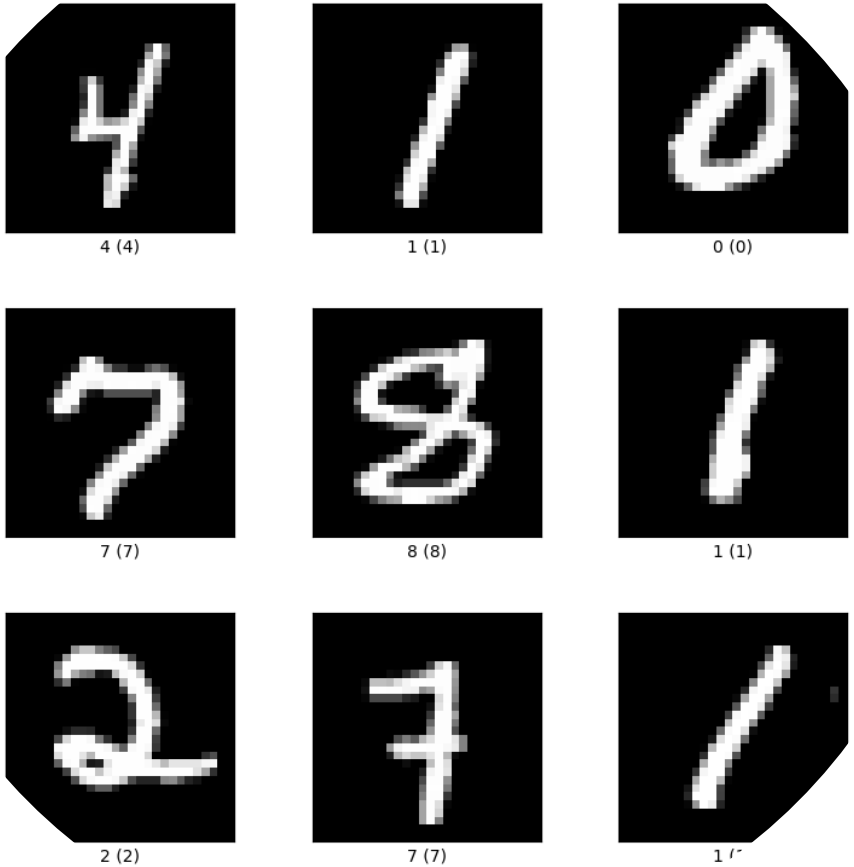
For planning I used a Gantt chart. I made it very specific in order to prevent myself from procrastinating until deadlines. This method of time management was quite effective.

Research

Theory	Coding
"Computer vision with python" by Eric Solem	Free resources like W3Schools
YouTube videos such as the series by 3Blue1Brown	Official documentation for libraries like NumPy
Medium.com articles	Other people's projects on GitHub
My brother's data science lecture slides	My brother's data science lecture slides





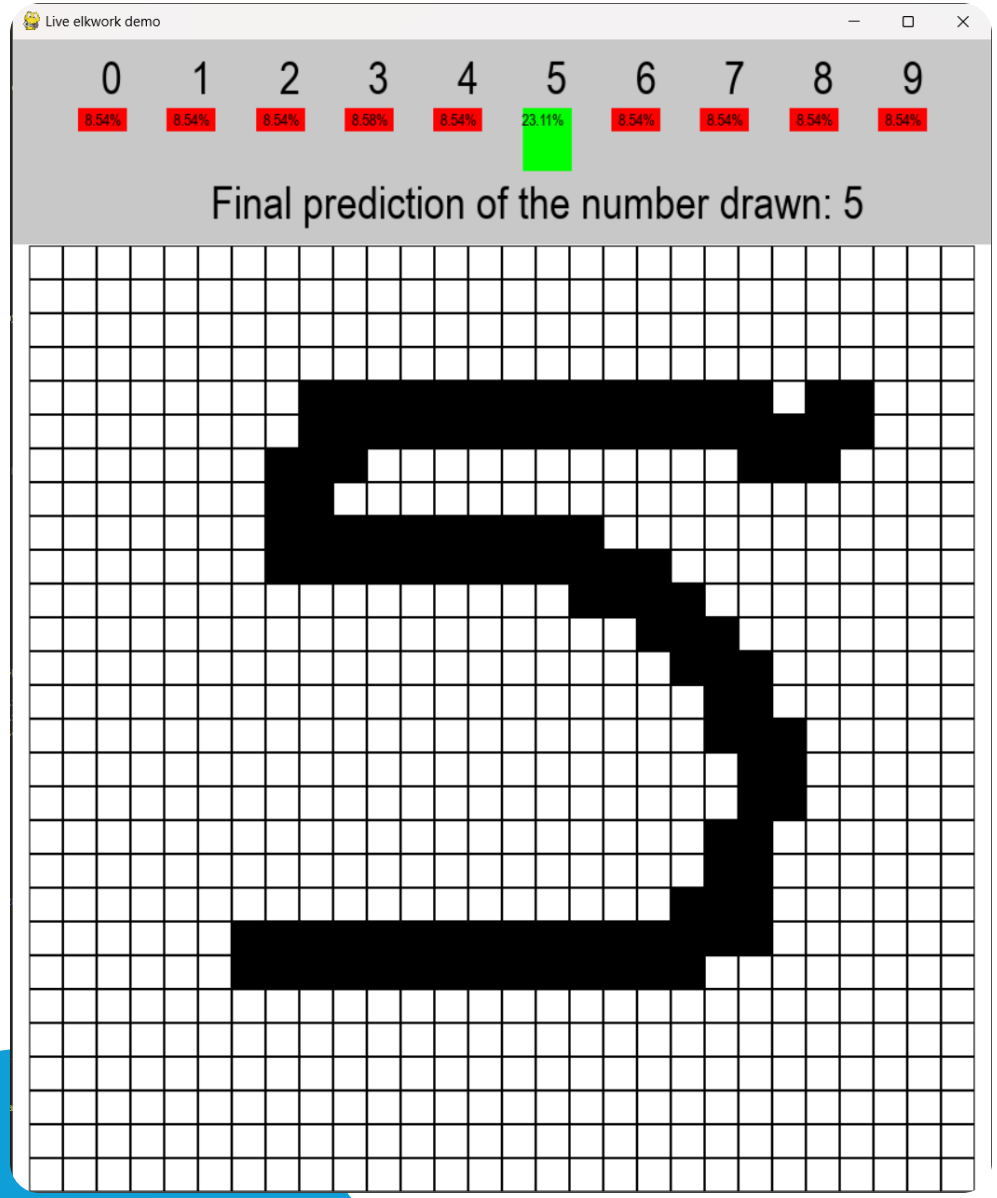


Main dataset used for this project.

Simple 28*28 image recognition dataset – prevents me being limited by my computer being weak.

The process

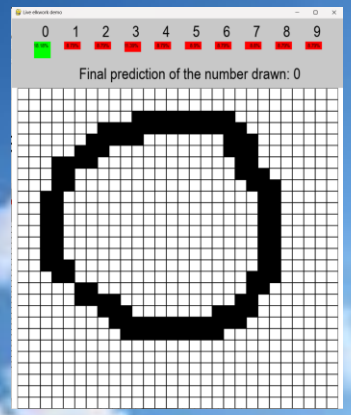
- I chose the language Python as I am very familiar with it.
- For documentation I first started using Word however switched to LaTeX half way though because I wanted to learn it.
- I used VSCode as my code editor.
- All math was done through NumPy for faster computation and all graphs were plotted through MatPlotLib.

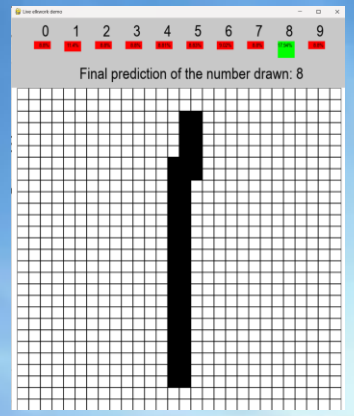


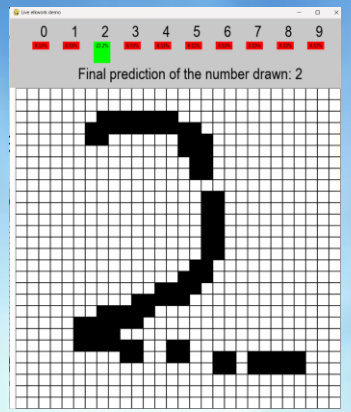
The product

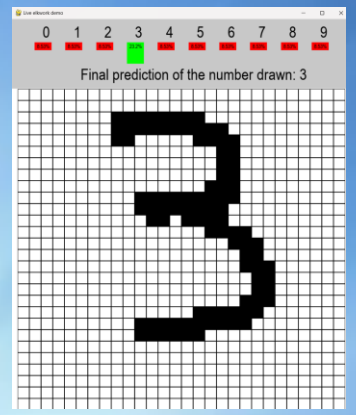
- The final product was a set of four Python files in which were stored:
- The network itself
- Graph plotting functions
- Parameter tweaking functions
- An interactive program to allow users to draw their own images.

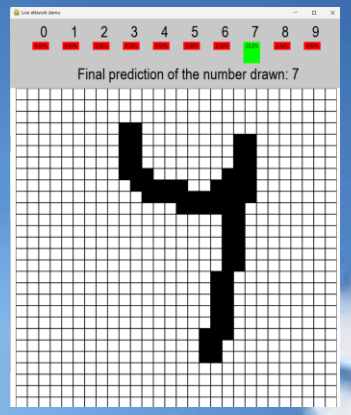
LIVE DEMONSTRATION

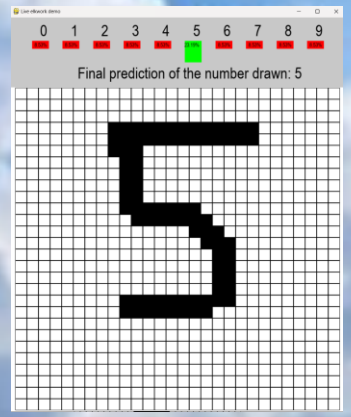


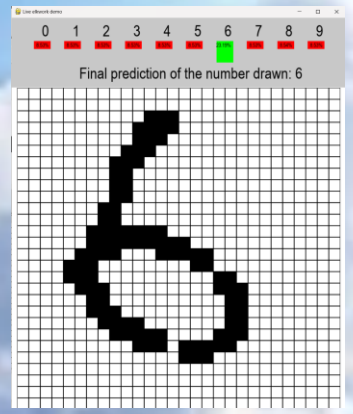


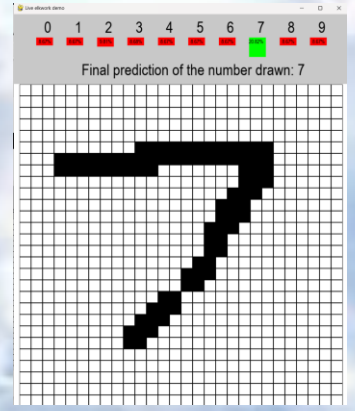


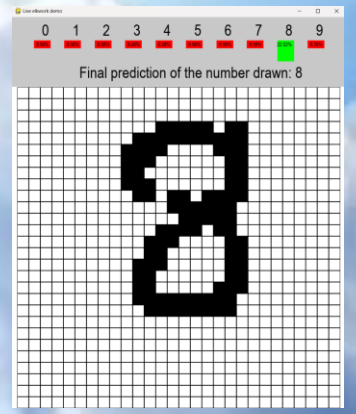


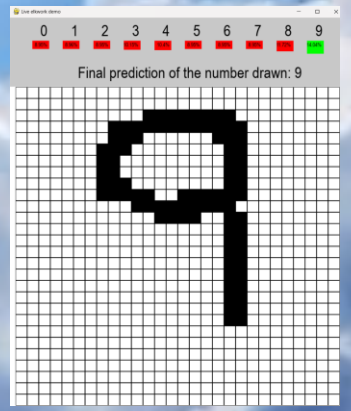












Successes

MNIST dataset item

FashionMNIST dataset item

CIFAR10 dataset item

Made my own machine learning library

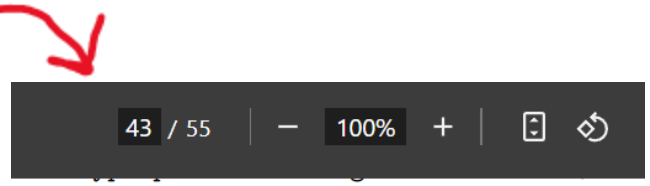
Made an optimized neural network model to recognise the MNIST dataset at 98.51%, FashionMNIST on 89% and CIFAR10 on 60%

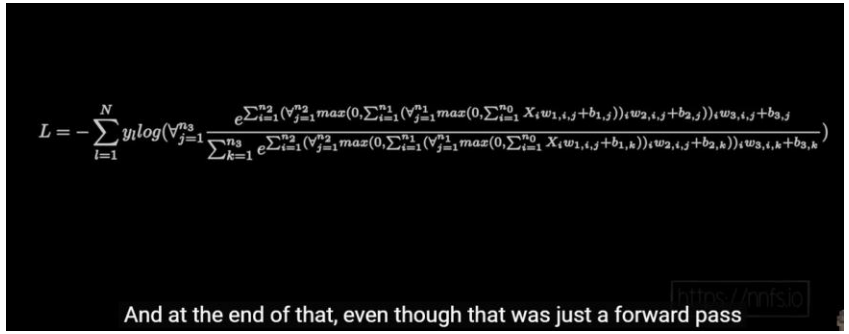
Consequently, created a highly optimized model that has a higher accuracy than every example I could find on the internet (9.4% more than highest)

Library import: **pip install elkwork**
Source code can be found on
Leonid-Elkin.github.io

50	**Convolutional Clustering**	1.4		Convolutional Clustering for Unsupervised Learning		2015
51	**CNN Model by Som**	1.41	98.59	Convolutional Sequence to Sequence Learning		2022
52	**Weighted Tsetlin Machine**	1.5	98.5	The Weighted Tsetlin Machine: Compressed Representations with Weighted Clauses		2019 TM
53	**MLP** (ideal number of groups)	1.67		On the Ideal Number of Groups for Isometric Gradient Propagation		2023

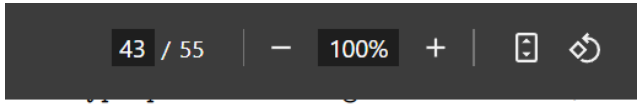
Failures





- Although all project objectives met, time management was extremely poor.
- Severe delays in progress were encountered due to unexpected events.
- Initial project plan was very ambitious, and I was too stubborn to simplify it along the way.
- Even though this is an artefact, the writeup was 12500 words long, with 900 lines of python and 1800 of LaTeX. This is made even worse by the fact that I had to rewrite the entire word document into LaTeX after 7k words.
- Some aspects of the investigation are still not fully justified – further research and writeup should be done.

Areas I would do differently

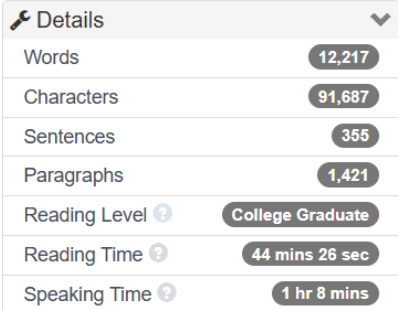


- Reduce workload by simplifying the project – Maybe focus on one particular factor.
- Plan code in advance
- Plan use of software in advance – this would have saved me rewriting 7k words when making the switch to LaTeX.
- Try and work ahead of deadlines to prevent myself from cramming work into a short period of time.

```
905        file.close()
906
907
```

```
1607 \bibliographystyle{EIIItr}
1608 \bibliography{CitationList}
1609
1610 \end{document}
1611
```

```
200   howpublished = {\url{https://paperswith
201   note         = {Accessed: 2025-04-28}
202 }
203
```



What did I learn

- This project improved my programming, especially working with objects and enormous data structures.
- Learned to use essential library such as matplotlib and NumPy.
- Learned LaTeX
- Learned important theory that will help me in university.

Thank you for watching