

MICRON NUS-ISE BUSINESS  
ANALYTICS CASE COMPETITION  
2022

# Team Microsoft

Cao Yuxuan, Jin Cheng Hao, Lee Jing Hong, Lim En Hao

Pre-university Category

Hwa Chong Institution

# Question 1

## Micron's Role

As a manufacturer who is also a dominant firm with a large say in the supply and distribution side of the production process, Micron's role in this chip shortage is to manage increasing demand and create a favourable environment where the shortage can be resolved.

Micron has invested heavily in its supply chain to make sure that there is redundancy in operations.<sup>1</sup> Their globally diversified production has also been very helpful in this regard because it mitigates the COVID-19 protocols that were impacting production in certain areas.

Specifically, as a major player in the semiconductor industry, Micron has now come up with a US\$150 billion, 10-year investment plan<sup>2</sup>. This includes potential US fab expansion as well as working with governments around the world including Europe where Micron does not have a manufacturing presence in. By doing so, Micron is able to be more self-sufficient. From a supply chain perspective, this means that Micron is able to have more say and control in the sourcing of raw materials and distribution stages of the production process, making it more resistant to supply shocks, therefore enabling it to help alleviate the shortage by injecting more supply into the market.

Micron plays a pivotal role in the retail sector as many of its major customers depend heavily on the performance of their physical stores. One example of which is Apple, in April 2021<sup>3</sup>, the production of iPhones and MacBooks suffered delays amongst the global memory chip shortage. This subsequently led to a slash in the production of Apple's latest iPhone 13 line, from the initial 90 million units to 80 million units.

## Impact on Micron

Global chip shortages have disrupted supply chains for complementary goods, specifically computing-purpose chips produced by other companies like Intel and AMD to be paired with DRAM and NAND chips to form a complete system. This not only affects the PC

---

<sup>1</sup><https://www.bloomberg.com/news/videos/2021-11-19/micron-ceo-mehrotra-global-chip-shortages-are-easing-video>

<sup>2</sup> <https://techhq.com/2021/10/micron-technology-and-its-us150-billion-bet-on-the-chip-supply-chain/>

<sup>3</sup> <https://pocketnow.com/ipad-and-macbook-are-the-latest-products-to-be-hit-by-the-global-supply-shortage>

and server markets, but also daily electronics like smart home devices and cars with entertainment or advanced electronic systems which use eMMC flash chips intensively. In fact, Tesla, a downstream user of Micron's memory chips, recently had to remove multiple in-cabin USB ports due to a lack of relevant computing chips for it<sup>4</sup>. Without computing chips, memory chips are merely flat pieces of silicon.

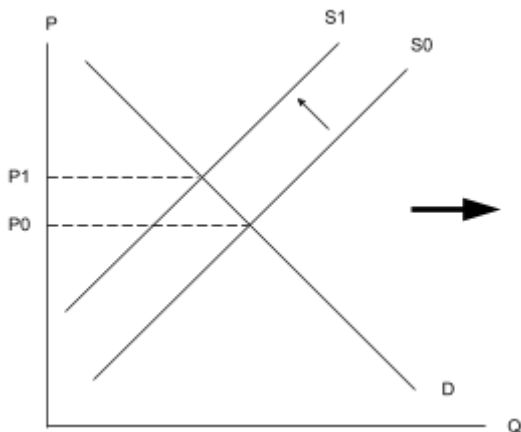


Figure 1: Market for complementary chips

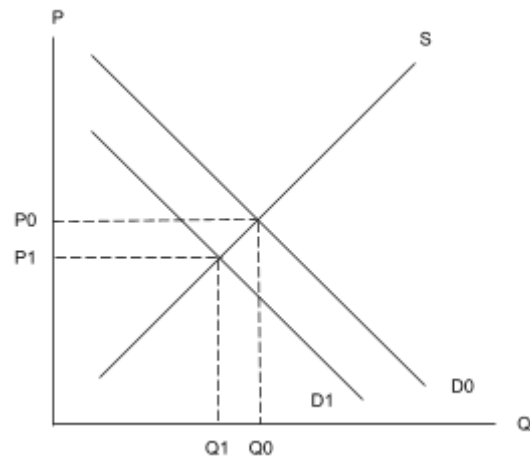


Figure 2: Market for DRAM and NAND.  
Revenue decreases from  $P_0Q_0$  to  $P_1Q_1$ .

Hence, shortages in supply of non-memory chips, a complementary good to memory chips, required for various technological products increases its price, leading to demand (and thus revenue) for DRAM and NAND chips falling correspondingly<sup>5</sup>.

However, shortages have also resulted in the DRAM chip prices surging rapidly. Given the heavy international reliance on memory chips for technological products fuelled by recent lockdowns/remote working conditions, demand for DRAM chips is likely price inelastic. Moreover, the uncertainty associated with shortage-induced price fluctuations and volatility leads to reduced price transparency,<sup>6</sup> further contributing to a lower price elasticity of demand. Hence, shortages in supply over a price inelastic demand lead to higher revenues

<sup>4</sup> <https://electrek.co/2021/11/12/new-teslas-delivered-with-surprise-missing-usb-ports-due-to-parts-shortages/>

<sup>5</sup> <https://www.electronicdesign.com/technologies/embedded-revolution/article/21176824/electronic-design-ongoing-pc-chip-shortage-takes-toll-on-micron>

<sup>6</sup> <https://eml.berkeley.edu/~dromer/papers/jmcb03.pdf>

for Micron despite specific shortages in complementary goods, as evidenced by data centre (70%), automotive (25%)<sup>7</sup> and DRAM revenue growing consistently.

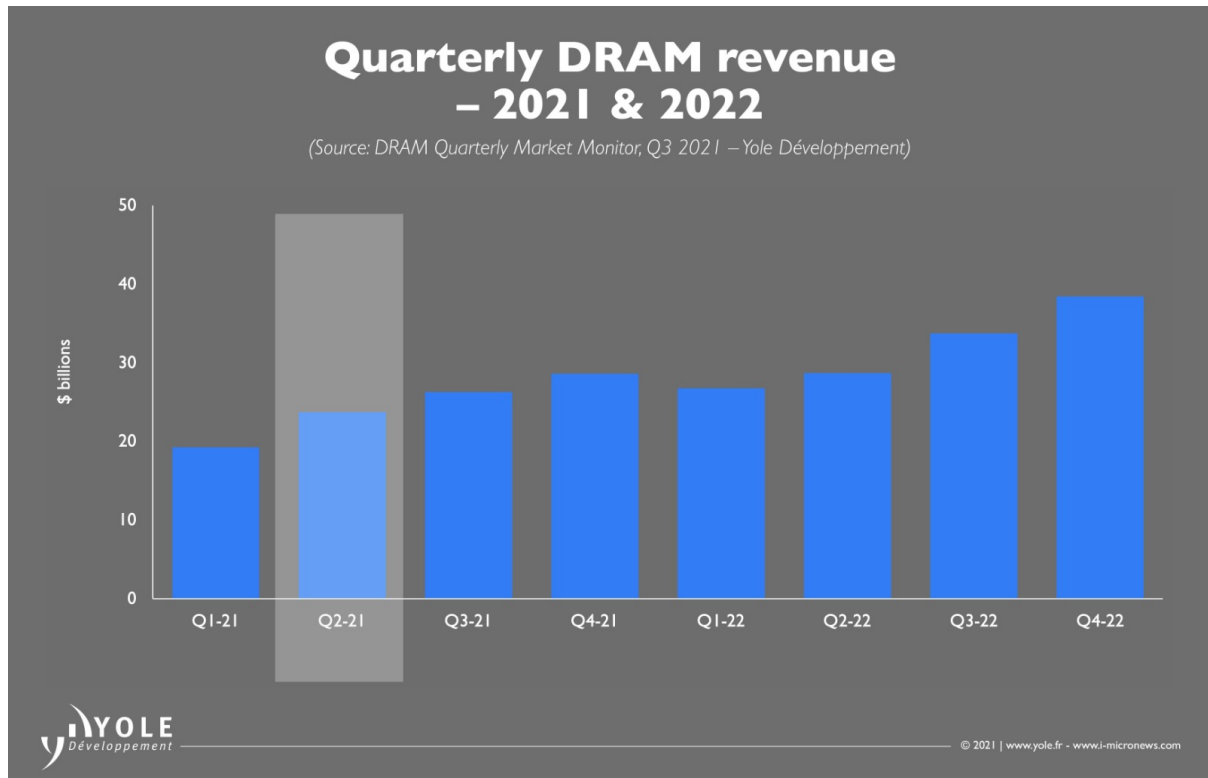


Figure 3: Quarterly DRAM revenue projection by Yole Développement

Meanwhile, the cost of error for Micron and similar chip production companies has skyrocketed during this shortage, with disruptions incurring greater operational costs due to the need for better emergency planning and response. For example, the recent incident of contaminated flash chips from Western Digital factory. WD was forced to respond to sudden cost increases by raising the prices of its affected products.<sup>8</sup>

However, unlike most other semiconductor companies, Micron's business model is not fabless and is therefore less reliant on foundries such as Taiwan Semiconductor Manufacturing Company (TSMC), a chokepoint in the global supply chains.<sup>9</sup> Hence, Micron can avoid paying large premium sums<sup>10</sup> to reserve production capacity at foundries like TSMC, significantly reducing the impact on its cost of production during the global shortage. Micron is also able to acquire greater production capacity in advance such as extreme

<sup>7</sup> <https://mobile.reuters.com/article/amp/idUSKBN2IZ1ZY>

<sup>8</sup> <https://inf.news/en/digital/cf14a5d39255e1fa27bfe0571e0b27ef.html>

<sup>9</sup> <https://www.bloomberg.com/news/features/2021-01-25/the-world-is-dangerously-dependent-on-taiwan-for-semiconductors>

<sup>10</sup> <https://www.hardwaretimes.com/nvidia-pays-tsmc-7-billion-to-secure-5nm-chips-for-rtx-4080-4090-gpus/>

ultraviolet lithography machines from dutch firm ASML<sup>11</sup> in response to imminent shortages. Micron is therefore well positioned to counteract potential increases in cost of error.

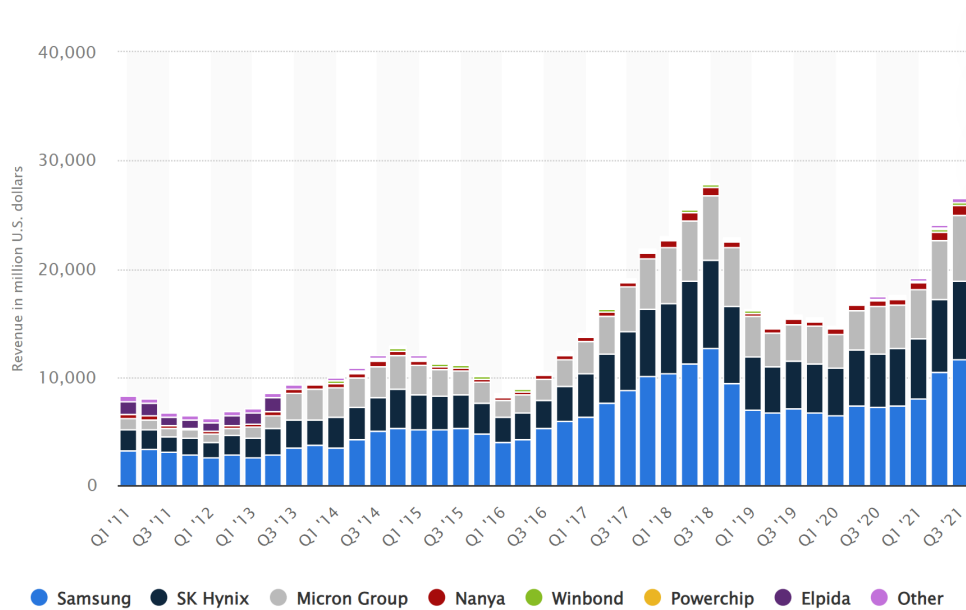
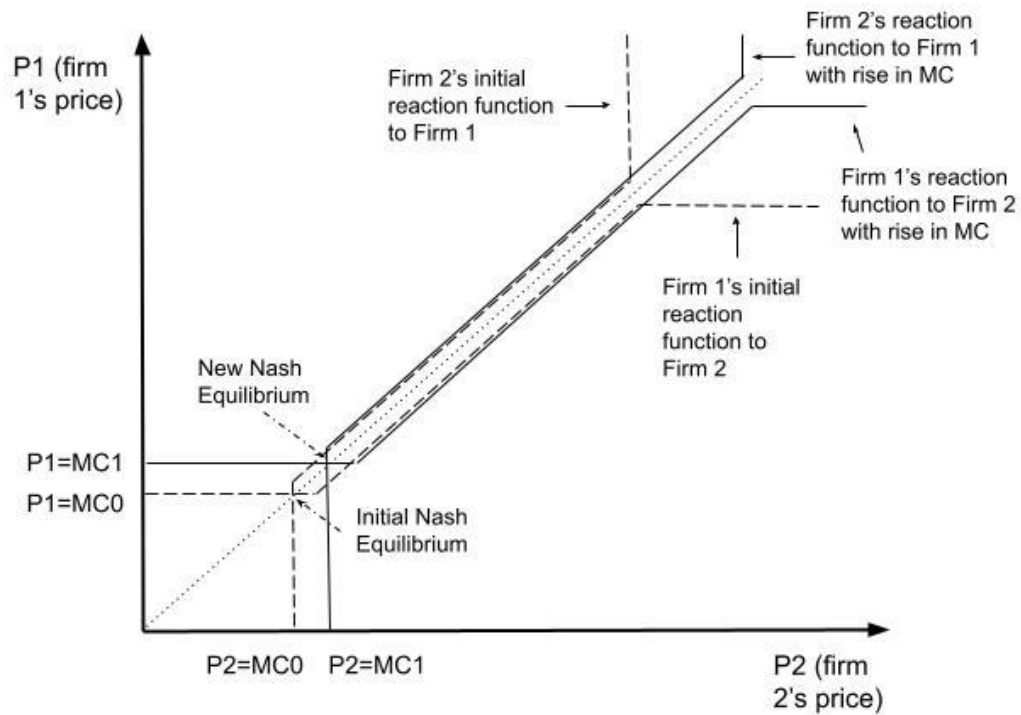


Figure 4: DRAM manufacturers revenue worldwide from 2011 to 2021, by quarter via Statista<sup>12</sup>

Additionally, given the DRAM industry's oligopolistic market structure (i.e. market dominance by Micron, Samsung, Sk Hynix), uniform increases in costs leads to higher prices, and by extension, higher revenues.

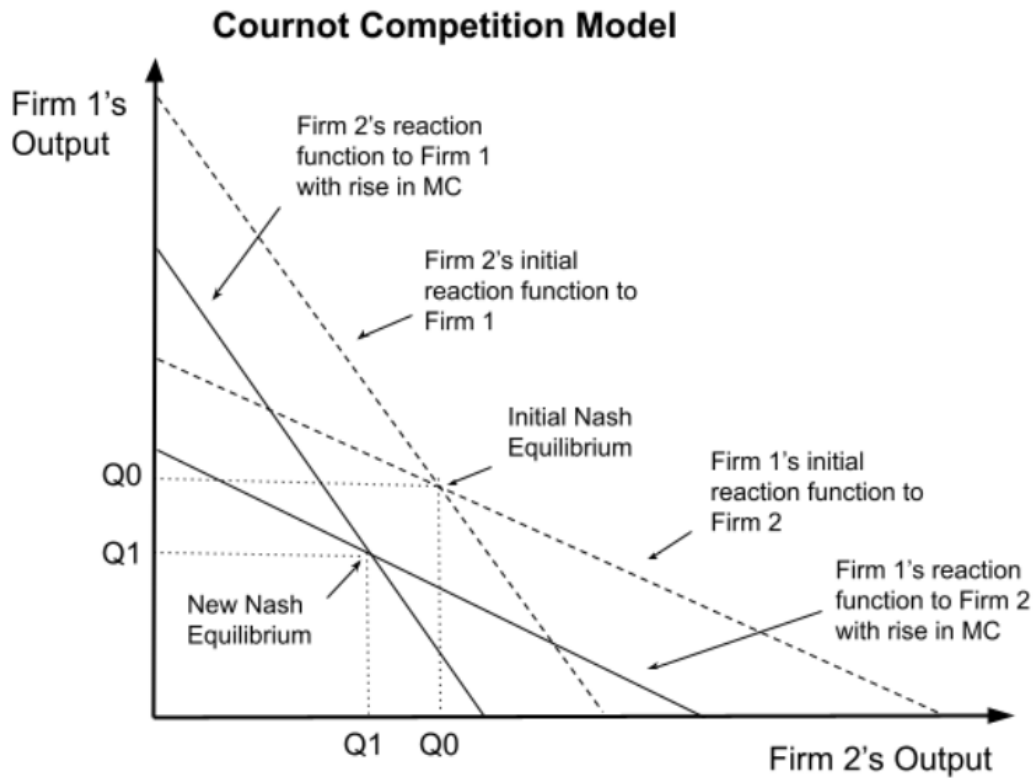
<sup>11</sup> <https://siliconangle.com/2021/06/30/micron-says-demand-memory-chips-will-stay-strong-stock-falls-anyway/>

<sup>12</sup> <https://www.statista.com/statistics/271725/global-dram-revenues-of-semiconductor-companies-since-2010/>

**Bertrand Competition Model**

*Figure 5: The Bertrand competition model*

Utilising the Bertrand competition model, an increase in MC shifts the respective firms' Reaction Function (RF), obtaining a Nash equilibrium of higher prices.



*Figure 6: The Cournot competition model*

The Cournot competition model displays similar outcomes - uniformly higher MC lowers their respective RFs, leading to lower output and by Law of Demand, higher prices. Thus, even if the cost of error increases, this likely results in a net increase of profits for oligopolists like Micron.

## Question 2.1 - Demand Forecasting<sup>13</sup>

We first approached this question from a mathematical standpoint. From inspection, we could observe that the curve had a clear period of around 60 weeks and a simple upwards trend. Therefore, we developed the following mathematical model to approximate this:

$$P(x) = ax - k \left[ \cos\left(\frac{2\pi}{p}x\right) + \cos\left(\frac{2\pi}{p}\right) \right] + E$$

where:

$P(x)$  : prediction function, where the argument is the # of the week being predicted,

$a$  = gradient of slope,

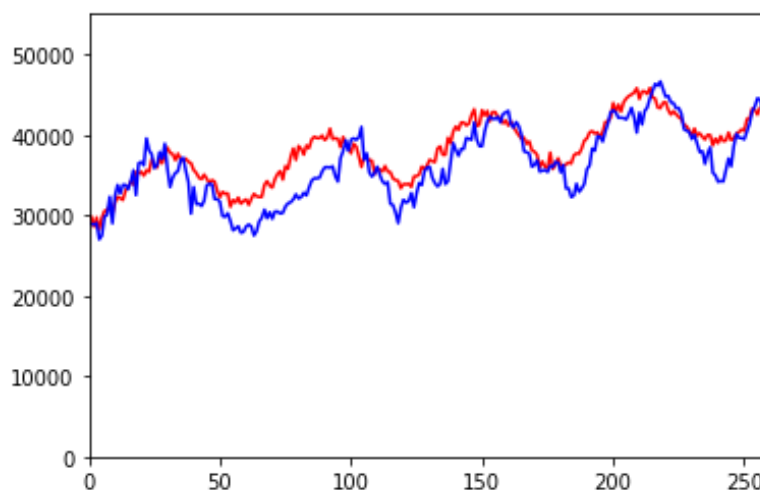
$k$  = amplitude of curve,

$p$  = period of curve,

$E$  = random variable that follows a Gaussian distribution defined by the standard deviation of error terms between the given values and their moving average.

In this method, we used a moving average to smooth the curve and derive the amplitude from the average peak-trough distance of each period. By subtracting a backshifted window of time from the current timestep, we were also able to obtain a trendline, albeit a very noisy one. The detailed calculations can be found in this notebook.<sup>14</sup>

The graph of the prediction as plotted against the given data gives a mean absolute error (MAE) of around **2200**.



(Blue line = given data, red line =  $P(x)$ )

<sup>13</sup> The full code we used for this question can be found at <https://github.com/aliencaocao/BACC-2022>

<sup>14</sup> <https://github.com/aliencaocao/BACC-2022/blob/master/maths%20model.ipynb>



The main limitation of this method is that the demand curve is not perfectly analogous to a cosine curve, as the troughs are sharper while the peaks are rounder. Therefore, it can be seen that the graph is able to predict the peaks slightly more accurately.

We realised that just by using naive methods like the one described above, we are able to model seasonal trends well, but it is obviously not flexible enough and will become increasingly unreliable as time goes by and the cyclic pattern and numerical values drift from our fixated function slowly. Thus, we directed our attention to deep learning, an increasingly popular and powerful data analytics tool. Inspired by the universal approximation theorem, we hypothesize that Neural Networks are able to adapt to the given dataset better due to its large information receptive field and high degree of non-linearity, driven by its high parameter count. Furthermore, neural networks can be retrained on new data in the future, allowing them to constantly improve themselves automatically and adapt to new trends. This is not possible with mathematical functions modelling and manually adjusting the mathematical function is tedious.

We used the TensorFlow deep learning framework by Google, and various other Python libraries like Pandas and Matplotlib to help us improve our model. Everything was implemented in Python 3.

Initially, we treated it as a classic time series forecasting problem in the fields of machine learning (ML). We architected a model based on Recurrent Neural Networks (RNNs)<sup>15</sup> where we provide the model with a variable length of input (in this case we fixed it to 52 timesteps/weeks), and the model predicts the demand for the next timestep/week. Specifically:  $y = f(x)$  where  $y$  is the demand for next week,  $x$  is a length  $N$  vector of the past  $N$  weeks' demand **in sequence**.  $f(x)$  is the model to be learnt.

RNNs are suitable for time series forecasting as they can learn sequential patterns well and have flexibility in both the input and output length. In our case, we used a variant of RNNs - Long Short Term Memory (LSTM),<sup>16</sup> aided with Spatial1D Dropouts and Layer Normalisation to reduce overfitting and stabilise training, with a 1D Convolutional layer at

---

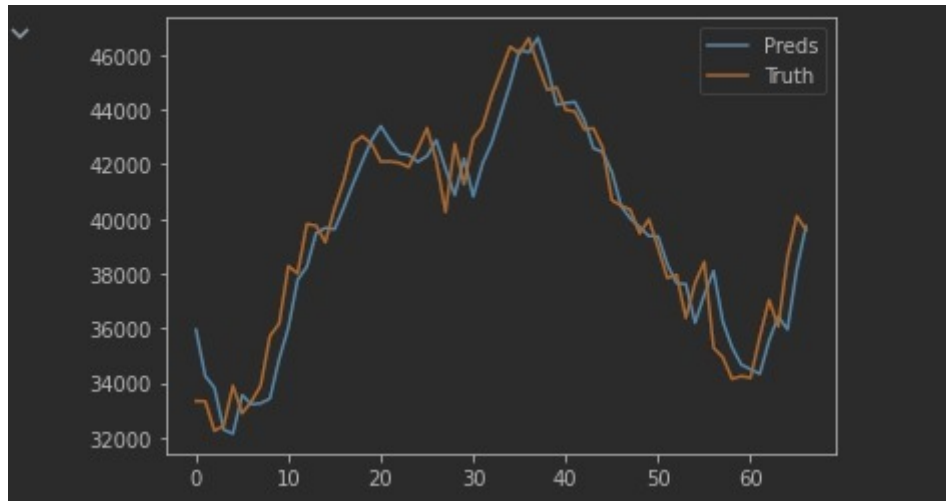
<sup>15</sup> <https://www.nature.com/articles/323533a0>

<sup>16</sup> <https://direct.mit.edu/neco/article-abstract/9/8/1735/6109/Long-Short-Term-Memory>

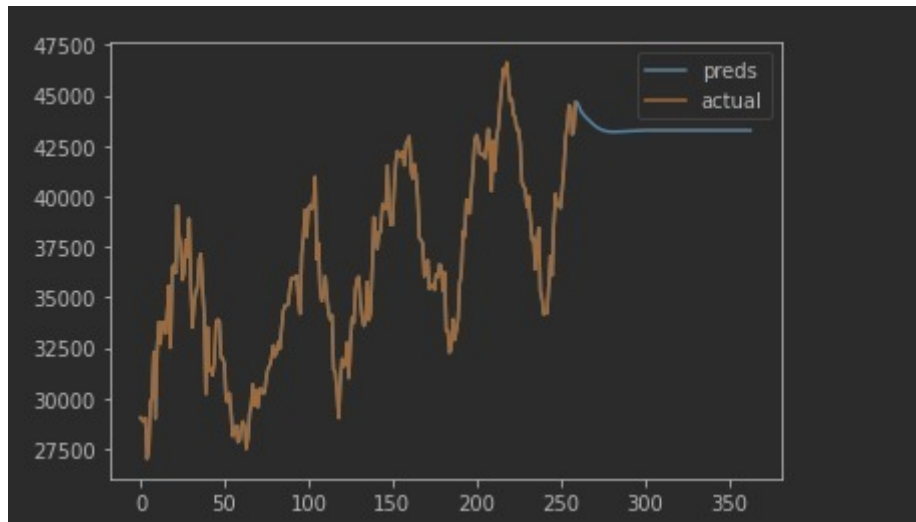
the top to learn inductive biases on the data. An overview of our RNN network architecture can be found in Appendix A.

We chose  $N$  to be 52. The assumption made by us at this stage was that 1 year/52 weeks can represent a full cycle of the trend for the model to learn (which we later discovered to be slightly off). Nevertheless, we trained the model with 4 years of data ( $52 \times 4$  weeks) and validated its performance on the remaining 52 weeks of data, and we were able to obtain MAE of **970** over the validation set.

However, after plotting and inspecting our results, we found out that the model seems to be misled and instead of learning to actually forecast values, it went to copy and paste the value of the previous timestep to the next.



As a result, the actual prediction on unseen data is not desirable and tends to be a straight line.



We believe that this is caused by both the simplicity of the dataset and the loss function that we are using. The dataset is highly seasonal and involves large portions of relatively linear relationship (e.g. prolonged period of constant upward/downward trend and lack of large 'zig-zag' patterns). As we used the MAE as our loss function, this can force the model into learning to copy the value from the previous time step only, as doing this will give the lowest MAE overall, due to the largely same trend between the copied previous timestep and the truth. Therefore, our attempt at RNNs is not satisfactory, even though it **seems** to have double the performance in terms of MAE as the naive method.

We hypothesise that for us to prevent a model from learning such 'copying' tricks, our input has to have no easily-learnt sequential relation to the target value, i.e. (a, b, c, d) cannot have a relationship where  $a+b=c$ ,  $b+c=d$  etc. This may seem counterintuitive to us as we would think that the value that is the most recent should have the highest importance on determining the next value of a series. While this is true, the model will find that the importance is so large that it starts to disregard all the previous values and just focus on the immediate previous one, resulting in the 'copying' behaviour.

To mitigate this issue, we can feed the historical data in a way that can be treated equally by the model, thus we decided to adopt a more basic way of modelling relationships - regression, where every timestep inputted matters equally to the model (or at least on the surface, before it actually learns which are more important). To do this, we architected a Multi-layer Perceptron (MLP) model, also known as a Deep Neural Network (DNN) using fully connected (FC) layers. In between each FC layer, we used the Mish activation function

where  $mish(x) = x \tanh(\ln(1 + e^x))$  to increase non-linearity of the model. The activation function is something that will be applied to every output of an FC layer before it enters the next layer. This resulted in a model with 49,281 parameters. As suggested by the author of Mish, we used the Ranger optimizer for gradient descent, which is essentially a Rectified Adam<sup>17</sup> optimizer with Look Ahead mechanism<sup>18</sup>.



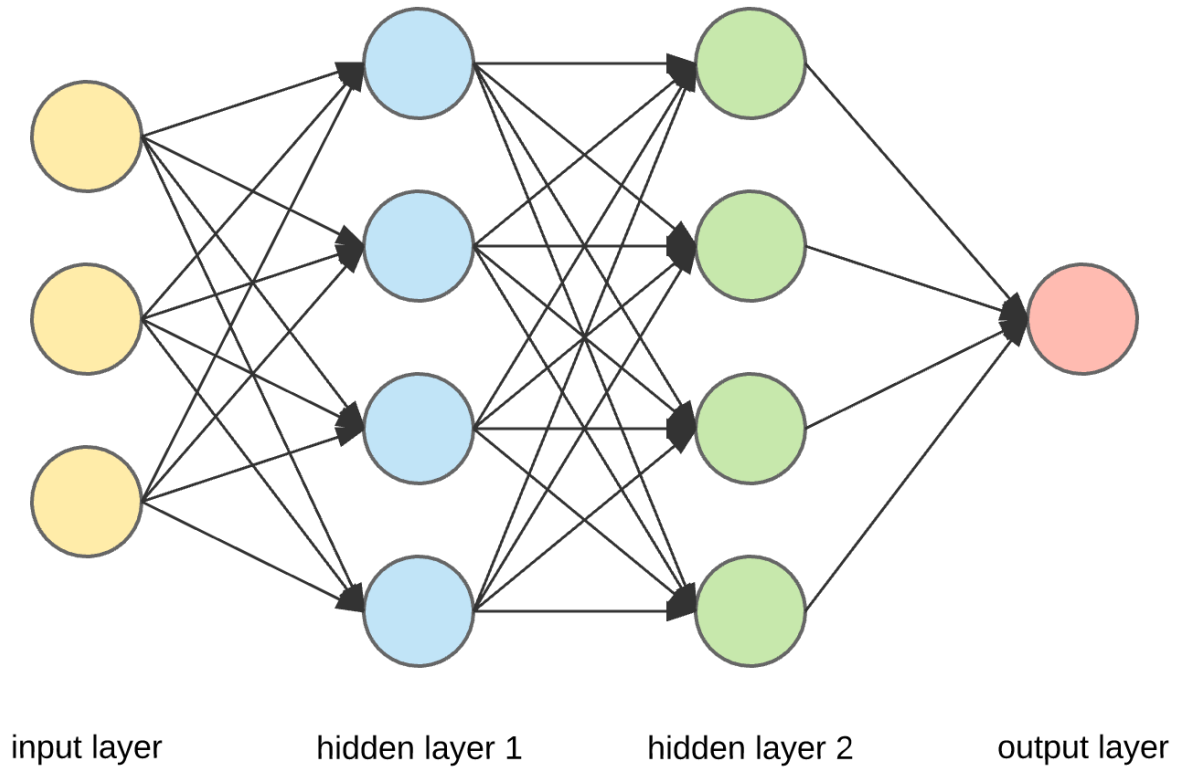
An overview of our MLP network architecture

Similar to our first attempt at RNNs, we formulate the problem as follows:  $y = f(x)$  where  $y$  is the demand for next week,  $x$  is a  $1 \times N$  vector of past  $N$  week's demand (sequence is **not** considered).  $f(x)$  is the model to be learnt.

A single FC layer can be understood as a linear vector function of  $y = w \cdot x + b$  where  $w$  is the learnt weight vector,  $x$  is the input vector (in this case,  $1 \times N$ ) and  $b$  is the bias to be added element-wise onto the dot product of  $w$  and  $x$ . The output  $y$  is then being fed into each and every neuron in the next FC layer as one of the values in the input vector  $x$ .

<sup>17</sup> <https://arxiv.org/abs/1908.03265>

<sup>18</sup> <https://arxiv.org/abs/1907.08610>

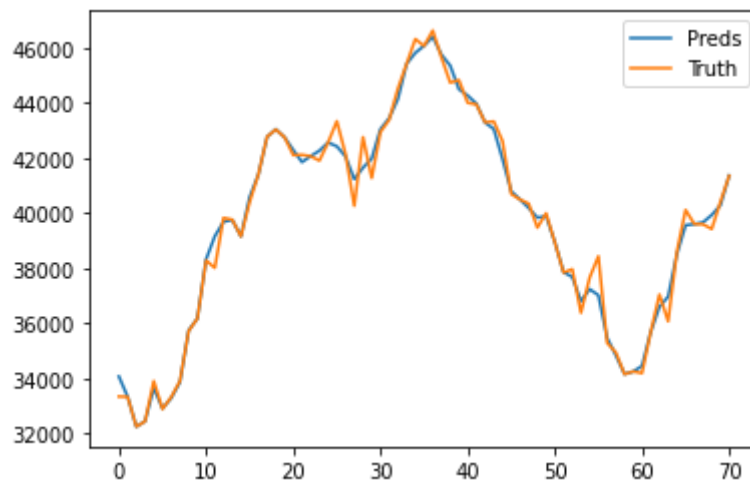


Note that in this case, we have to carefully choose the value for  $N$ , as for MLPs, the input size must be specified and fixed, and it greatly affects the model quality. Theoretically, the more 'features' you have/the more historical data you feed at once, the more accurate your prediction will be. We indeed observed this trend during the tuning process, however, it is not feasible and optimal to use the highest number of features we can afford to use - all past 259 weeks of data, since there will only be 1 effective pair of training sample, which is far from enough. During the process of choosing  $N$ , we also observed that the period of the trend in demand is actually longer than 1 year/52 weeks and varies slightly. On average, the trend period was 58 weeks, with the longest being 66 weeks and the shortest being 54 weeks. We used the gap between each minimum point to calculate the period. We also observed that the starting week for the first period is around week 5 of 2017. After balancing and experimentation, we decided to go with using roughly past 3 periods as input and still predict the week ahead as output.

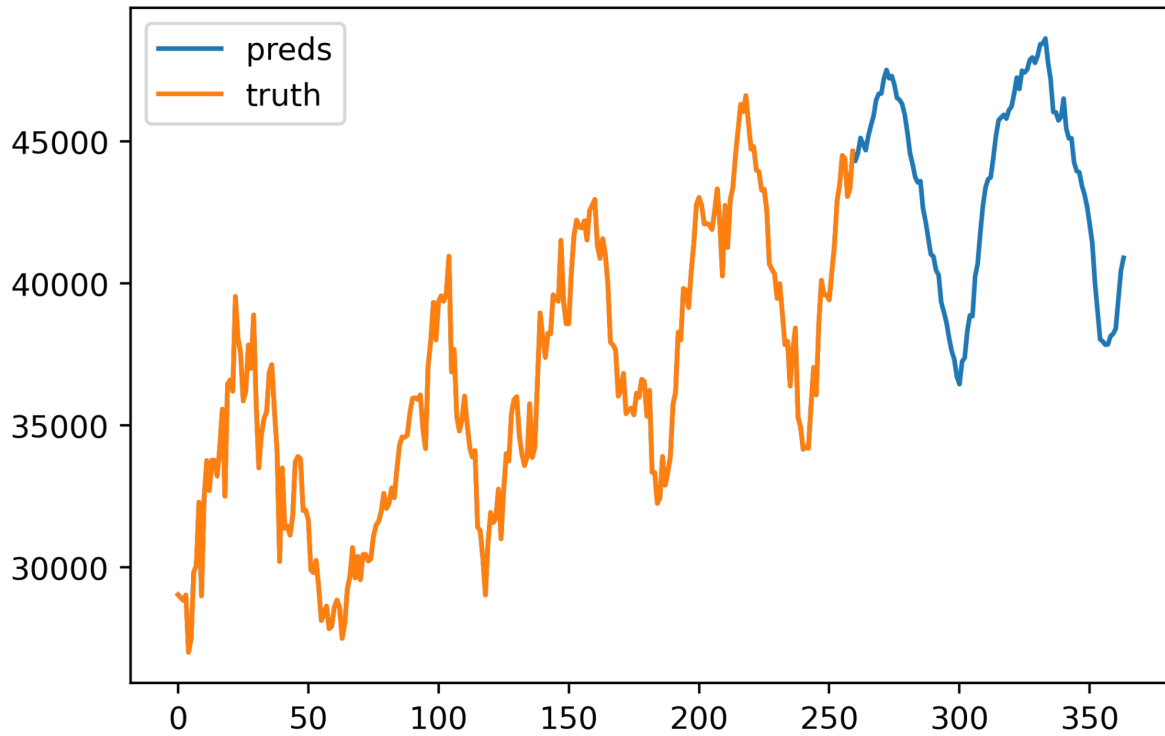
Due to the stochastic nature of deep learning, we trained 2 different models using slightly different  $N$  values:  $52 \times 3 = 156$  and  $58 \times 3 = 174$ . We then ensemble both models and take the average output between them as our final prediction. Doing so reduces the output

noise ('zig-zag lines') as ideally, we would want a smoother projection to aid with decision making, even though the reality may be very noisy.

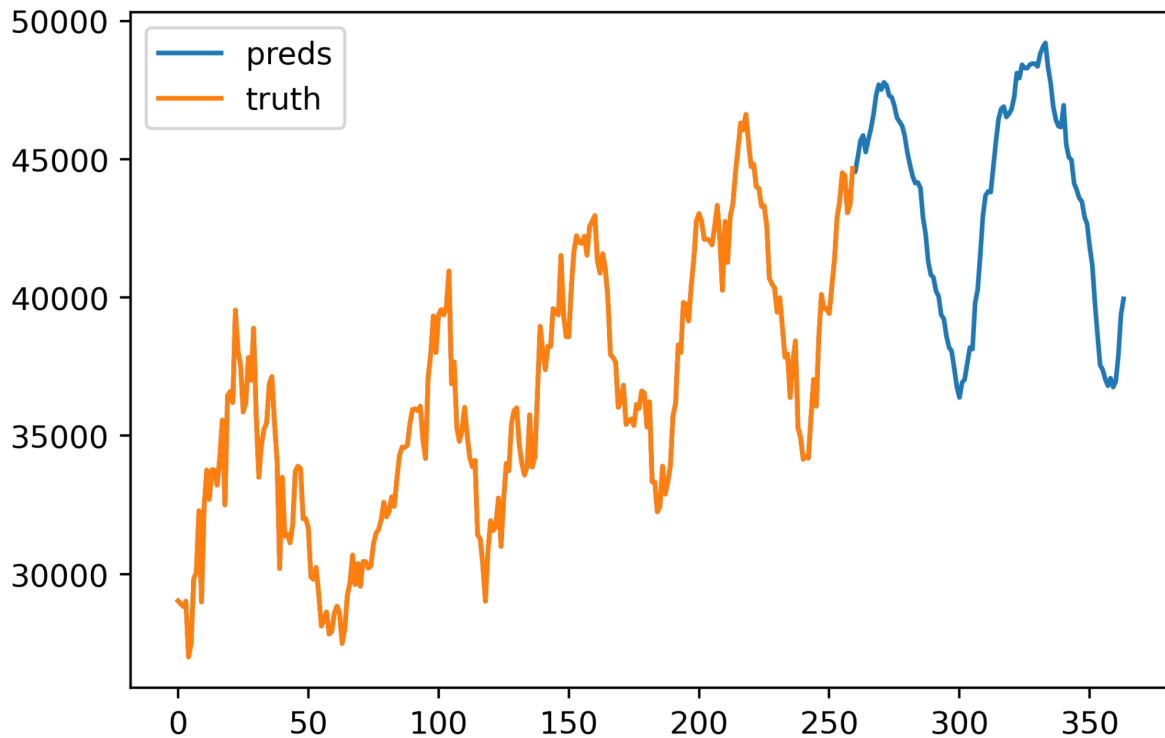
We first validated and tuned our model architecture and hyperparameters using the remaining 86 weeks as the validation set. Once we are satisfied with the model's performance, we retrained the model using the entire dataset of 260 weeks without a validation set so that it can learn the latest trends from the latest data. Eventually, we are able to obtain a satisfactory MAE of **244** on the training data (and 269 for N=156), a near 4 times improvement over the RNN model. The 'copying' issue is also greatly improved as shown by the graphs.



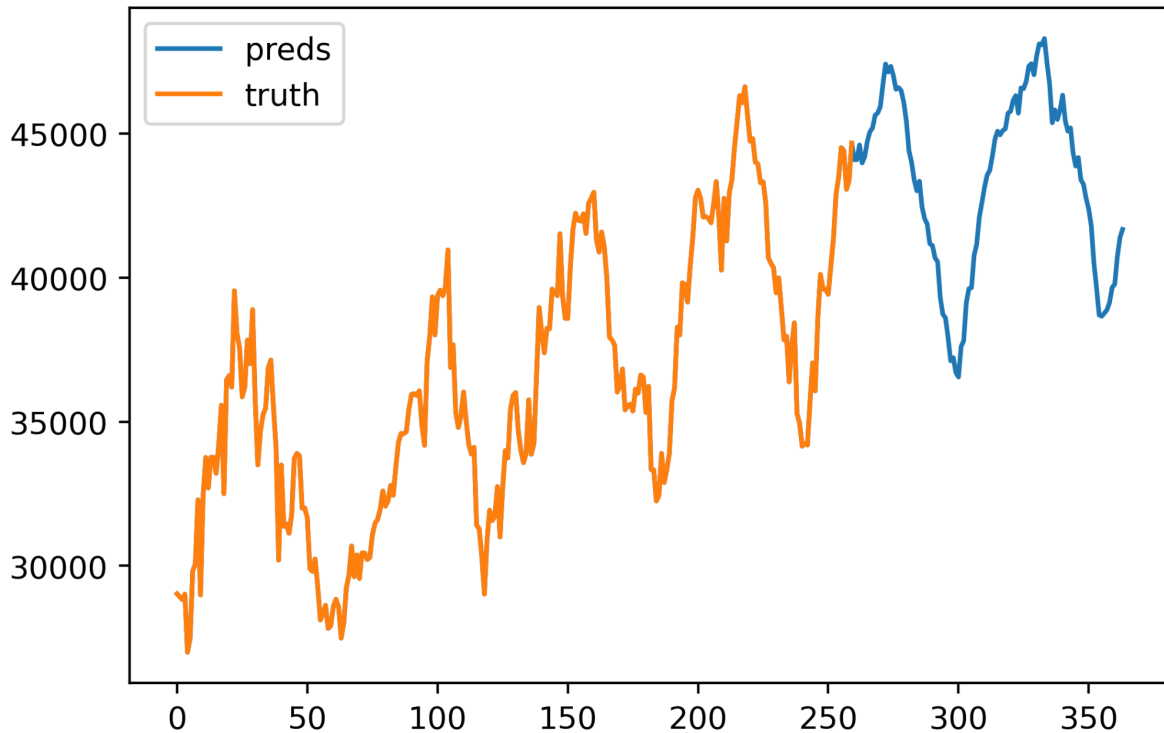
To verify that it actually learns, we ran predictions for 2 years/104 weeks into the future and obtained reasonable results. We can see that the model is able to capture information to a much greater detail than our previous attempts, while still learning the seasonal trend and upward trend well. For example, the model learnt from small spikes around weeks 45, 110, 175, 235, and predicted a similar spike around weeks 280 and 330.



Above is the result of the ensemble model. We went to verify that ensembling is effective in reducing noise as shown below.



Above are the predictions for the MLP model with N=174.



Above are the predictions for the MLP model with  $N=156$ .

It can be seen that both models have a larger degree of noise ('zig-zag' lines) in their output compared to the ensembled model, proving its effectiveness.

Although our MLP model is able to achieve exceptional performance, it also has its limitations compared to traditional mathematic models. The most obvious one being its 'black box' nature a.k.a. lack of explainability. Due to the number of parameters and weights of our model, it is impossible for humans to understand how the model arrives at conclusions, and thus it provides little to no value in studying the data and drawing insights from it from a human perspective. Another one is the nature of stochastic gradient descent - randomness and 'luck'. Each time the model is trained, even if on exact same data with exact same hyperparameters, the results could vary slightly. Although we employed many techniques like model ensembling to improve the reproducibility of forecasts, it is still not going to be 100% reproducible. This causes a small degree of uncertainty each time the model is being trained.

Besides the common limitations of all deep learning models as stated above, our MLP-approach has another limitation that affects its reusability. As mentioned above, the input dimensions for MLPs are fixed when the model is trained, and the performance of the



model scales proportionally with the number of features/# of timesteps fed in. This means that for other products with less data, this model architecture cannot be reused directly unless one modifies the input dimensions, which will then greatly impact the model's performance. In cases where the amount of data available is greatly limited, the model may not be able to learn much and may underperform the traditional mathematical approach, which is much less sensitive to the amount of data available. In the context of this task, this issue is negligible as we do have enough data for effective learning, but in the real world context, it is worth noting.

Out of interest, we attempted to explore further into using external data to aid us in prediction, but that is based on the assumption that the demand data given to us is somewhat representative of the actual trend seen by Micron and the dates are real, which we verified to a certain extent in Appendix B. However, due to the official disclaimer that the data given is arbitrary and not related to Micron products, we will not include this discussion here in the formal report.

## Question 2.2 - Minimum Machine Requirement<sup>19</sup>:

Upon initial examination, it would seem that in order to derive the number of machines required to meet the weekly forecasted demand, one would have to compute the machine capability/capacity for all 130 steps sequentially. However, not only is this unnecessarily tedious, but also unfeasible given that this drastically increases Raw Processing Time (RPT), substantially lowering machine capacity. Given that the number of machines required for Week  $N = \text{Demand}_{\text{Week } N} / \text{Machine Capacity}$ , a sequential approach significantly increases the number of machines required and thus reduces efficiency.

Hence, we adopted a macro, systems-level approach instead, where each type of machine operates simultaneously, completing the respective steps it is involved in. For instance, Albatross works on Step 48, 52, 55...127, and repeats this process after finishing Step 127. In the meantime, Cattle works on Step 4, 6, 8...122, repeating after reaching its last

<sup>19</sup>  Micron Case Competition - Q2.2 and 2.3

This file is also included in the supporting documents zip with the name "Micron Case Competition - Q2.2 and 2.3 .xlsx"

step, and so on. As such, instead of analysing this problem sequentially, we examined each machine type independently. We used the simplifying assumption that the RPT for each machine applied to the formulas stated is the summation of different RPTs for all steps that a specific machine is required in. This assumption is feasible and accurate given that machines operate simultaneously as explained above, not sequentially.

Thus, inputting the summation of different RPTs in the calculation of each machine's capacity gives their respective weekly output of products as follows:

Machine Specifications								
Machine Type	Albatross	Cattle	Dragon	Flamingo	Grouse	Herring	Peacock	Plovers
Total RPT	6,120.4	90.2	25.8	143.4	41.1	3,083.7	188.1	96.4
Utilisation (%)	80%	80%	70%	65%	69%	83%	73%	81%
RPT Basis	1	2	2	5	4	3	3	1
Load Size	200	1	1	1	1	50	1	1
Availability	90%	90%	80%	75%	79%	93%	83%	91%
Machine Capacity	237.16	160.87	437.58	171.34	534.76	378.48	97.41	77.07
No. of Machines	191	282	93	221	75	440	553	595

Once each machine's respective capacities have been obtained, we computed the number of machines of each type required to support our forecasted weekly demand in 2.1. If the number of machines required of a particular type is less than or equal to the initial number of machines available, it is highlighted in green, else it is highlighted in red. This provides a clear, overall picture of the number of machines required to support the forecasted weekly demand and serves as a guide to analysing Micron's ability to meet forecasted demand in 2.3.

## Question 2.3 - Optimising ability to meet demand<sup>20</sup>:

As seen from the previous section, we have calculated the minimum number of machines of each type that Micron requires in any given week to fulfill the demand projected

<sup>20</sup>  Micron Case Competition - Q2.2 and 2.3

This file is also included in the supporting documents zip with the name "Micron Case Competition - Q2.2 and 2.3 .xlsx"

for that week. In order to fulfill the demand for a particular week, Micron must have at least as many machines as is required to produce the demand for all 8 machine types, failing which the demand for that week will not be met due to a bottleneck in the machines that cannot produce enough. Assuming first that the number of machines will not change (i.e. Micron does not buy new machines), we see that Micron **cannot** possibly meet the demand between W1 and W37 (denoted by a thin black line in the sheet), as (1) the existing number of Flamingo machines cannot produce enough and (2) the order + installation time for Flamingos is longer than this 37-week period for which we have a production capacity deficit, which thus caps the total production of chips. For the subsequent weeks, however, we see another peak in demand which necessitates the purchase of machines. In order to solve this, we have two solutions.

- 1) Naive solution: refer to columns with ***bold and italic*** headers.

In this solution, we assume that the demand for any given week **must be produced in that week**. In that case, the number of machines that must be purchased is given by:

$$\max(\max(\text{demand}_{W1}, \dots, \text{demand}_{W104}) - \text{original number of machines}, 0)$$

The values for this are:

Albatross	Cattle	Dragon	Flamingo	Grouse	Herring	Peacock	Plovers
15	21	19	63	16	0	0	36

The optimal strategy in the naive case is thus to purchase the respective number of machines of each type in the first week. When these machines arrive (denoted by a thick black line in the sheet), we see that they are ready to face the next surge in demand as indicated by the second red portion.

- 2) Production-shifting solution: refer to all columns.

In this solution, we define F to be the first week in which a machine bought in W1 will be ready to operate. We define “production shifting” as the practice of producing batches that are **expected to be demanded** before they are demanded. The *difference* column is the original number of machines minus the number of machines required to meet the demand in that week. A positive number indicates a productive capacity surplus and vice versa. *New demand* is the adjusted production schedule. *Order Average* is the average required demand from F to the current week, where F is adjusted for production shifting if  $F > 38$ .

The reason why this model is preferable to the naive model is due to its flexibility and cost-reducing capabilities. Under the naive model, we must acquire exactly the number of machines that are *forecasted* to be required. In the production-shifting model, we can simply accumulate stockpiles of output that can be sold at later dates should the demand be lower than expected. Furthermore, such a model lends itself to the application of futures contracts, in which Micron can lock in future demand and therefore produce that demand now. In addition, the production-shifting model allows us to even out production, thus reducing the need to buy a large number of machines just to cope with a single week of abnormally high demand.

Recognize that under this model:

- Production can only ever be shifted earlier, not later. This is because you cannot retroactively sell to a consumer.
- Though we are still unable to meet the demand in W1-W37 due to a bottleneck of Flamingo machines, we are able to move forward the production of other machines that are **not bottlenecked** in order to reduce the number of such machines we must purchase at a later date.

Our model shifts demand in two steps.

- 1) Prior to F, we can produce a maximum number of batches as indicated by the current number of machines. Therefore, we can shift the surplus production from the lower red cells upwards into any green cell prior to F until it reaches the maximum number of batches.
  - a) Prior to this step, Herring and Peacock are within the initial production capacity. When this step is completed, we see that the demand for Albatross, Cattle, and Plovers can also be pre-produced in such a way that **no new machines are required during this 2-year period**. A sample of **one such production schedule** is provided under the column *New Demand* for those machines.
- 2) If after step 1 there is still a production deficit, we then subtract the sum of the surpluses within the *differences* column from the production in F and let it be the first figure under *Order Average*. This essentially means that all the deficit production capacity was taken from F **alone** and reallocated to the weeks with surplus capacity above F. After this, we compute *Order Average* for each of the weeks with a production deficit under the original schedule until we reach a week in which:

*Original # of machines required for that week < Order Average*

upon which we stop. After this, we let the last week in which the above condition does **not** hold be  $S$ , and we **randomly allocate** production capacity requirements of  $\text{ceil}(\text{order average of } S)$  and  $\text{floor}(\text{order average of } S)$  for **all** weeks between  $F$  and  $S$  inclusive, such that:

$$\text{Mean of new allocations} = \text{order average of } S$$

This ensures that the total number of machines required to fulfill all the demand is the same, just that some of the demand from later weeks is shifted forward in the production line so that productivity is maximized.

The mathematical reason why this works is as such: we can visualize the process of production-shifting as recursively moving production from week  $N+1$  to week  $N$  such that each week's values differ from an unknown optimal distribution by less than 1. We know that  $\exists M \in \mathbb{R}^+$  such that it represents the optimal distribution over a certain subset of weeks where we can theoretically use a fractional number of machines. When we reach week  $(S+1)$ , let us assume that it is still in the given subset of weeks; therefore:

$$|M - \# \text{ in } S| < 1 \ \& \ |M - \# \text{ in } (S+1)| < 1 \ \& \ \# \text{ in } S > \# \text{ in } (S+1).$$

Therefore, if we shift any positive integer amount of production from  $S+1$  to  $S$ , and we assume that  $|M - \# \text{ in } S| < 1$ , then it must be the case that  $M - (\# \text{ in } S) + 1 < 1$ , and thus:

$M - \# \text{ in } (S+1) - 1 < -1 < M - \# \text{ in } (S+1)$ . This is a contradiction, therefore week  $(S+1)$  is not in the given subset of weeks where  $M$  is the optimal distribution.

Another way to visualize this is by assuming that we add week  $(S+1)$  to the subset. In that case, as  $M > \# \text{ in } (S+1)$ , we will have to increase the output in week  $(S+1)$ . However, as we are **not changing the production of the following weeks**, and the sum of all output within 2 years must be the same, we violate the rule that production cannot be shifted to a later time.

Under this model, Micron should purchase the following number of machines in W1:

Albatross	Cattle	Dragon	Flamingo	Grouse	Herring	Peacock	Plovers
0	0	8	41	15	0	0	0

From the two calculations shown above, we recommend that the Micron management team follows the production-shifting model. Firstly, it means that less machines have to be

bought at the present time. This reduces fixed capital expenditure and frees up cash, while also reducing the amount of depreciation Micron's assets will undergo at the end of the year. Secondly, it means that production can be allocated much more efficiently. As seen in the graph, many of the weekly production capacities have not even been reached for certain machines, meaning that a spike in demand will not significantly affect Micron's ability to enact our second schedule. Lastly, even in the event that Micron wants to buy the recommended amount of machines as suggested by the naive solution, using the production-shifting method means that the purchases can be staggered instead of grouped in one order at the beginning of W1, which is likely to increase the real-world delivery time of each machine as Micron's suppliers might not be able to make so many machines at once.

## Question 3 - Linear Programming

We interpreted this question in two distinct ways. In the first interpretation, we assumed that machines X, Y and Z **must** work for **consecutive** hours and subsequently take a break in accordance with machine specifications given. In the second interpretation, which we solved prior to seeing the clarifications in the Q&A document, we assumed that machines **only** had to go for a break if they reached the **maximum** length of consecutive hours that they could work for. An example of this would be a machine of type X working for 21 hours, taking a 1 hour break, and working for another 21 hours. Given the possibility of both interpretations, we formulated the optimal solutions to both: the first model, which has since been confirmed to be the model by which we are judged, accounts for machines requiring recalibration after prolonged and continuous usage, while the second model accounts for machines overheating and taking short breaks in between running. We have included a theoretical proof and solution for the second model in Appendix C with code in Appendix D.

With regards to the first model, we define the following variables:

- $X$  = total number of machines of type X used
- $Y$  = total number of machines of type Y used
- $Z$  = total number of machines of type Z used
- $x_i$  = number of machines of type X used in the  $i$ -th shift,  $1 \leq i \leq 24$

- The  $i$ -th shift is defined as follows: A machine with consecutive operating period  $C$  and cooldown period  $D$  will work during *all* hours ( $H$ ) of a 24-hour day in which  $(i \leq H \leq i + C) \bmod (C+D)$ .
- For example, machines with the shift  $x_3$  will work from 3AM to 1AM of the next day (12AM - 1AM of the following day is considered hour #24). Machines with the shift  $y_4$  will work from 1AM to 2AM, 4AM to 2PM, and 4PM to 1AM the next day.
- Note that  $x_1 + \dots + x_{24} = X$ , and similarly for  $Y$  and  $Z$ .


Our objective function is as follows:

- Minimise  $104X + 90Y + 82Z$ , where  $X$ ,  $Y$ , and  $Z$  are non-negative integers. This is the summation of weighted costs associated with using each machine employed.

Our constraint functions consist of summing up all the  $x_i$ ,  $y_i$ , and  $z_i$  present for a particular hour. As this is too complicated to enumerate, a link to the Excel spreadsheet has been added in the footnotes.<sup>21</sup> The columns of the table in the “Production and Maintenance Schedule” represent each hour, while the rows represent the schedule of machines  $X$ ,  $Y$ , and/or  $Z$  working in that particular shift. For each column from 1 (AM) to 24 (1AM of the second day), the sum of all variables in that column must be greater than or equal to the number of machines required in that hour. The working shows a detailed breakdown of how many machines are required in each shift.

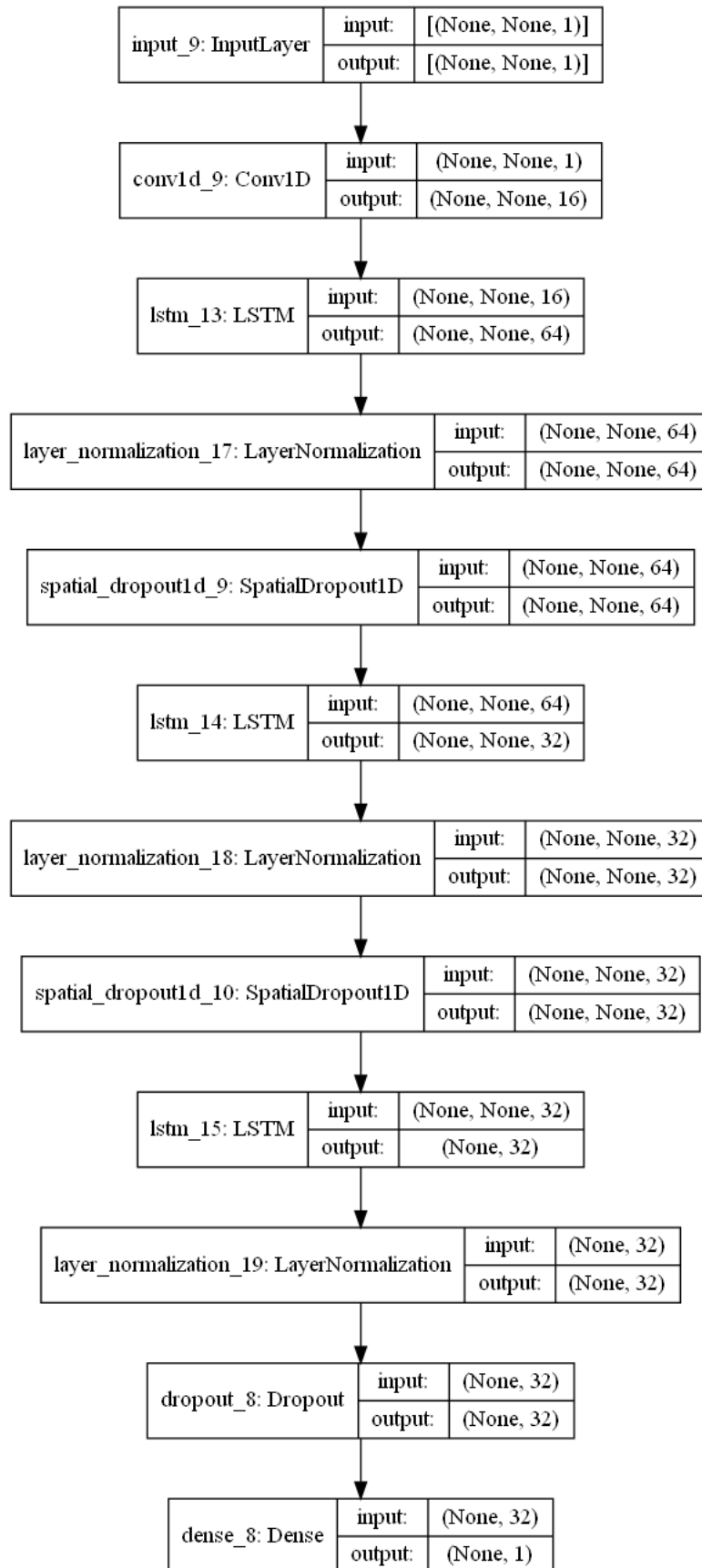
As such, our final answer would be: **53**  $X$ , **100**  $Y$ , and **86**  $Z$  machines are required. The total cost of this on a daily basis is **\$2,156,400**.

Intuitively, given that  $\text{Cost Efficiency} = \text{Operating Cost} / \text{Total Hours of Work completed in 24 hours}$ , this distribution makes sense as  $X$  is the least cost-efficient machine (\$472.72), followed by  $Z$  (\$455.55), and finally  $Y$  (\$450.00). Therefore,  $X$  is used the least, followed by  $Z$ , then  $Y$ .

<sup>21</sup>  Micron Case Competition - Q3 - Linear Programming & Optimisation Model

This file is also included in the supporting documents zip with the name “Micron Case Competition - Q3 - Linear Programming & Optimisation Model.xlsx”

## Appendix A:

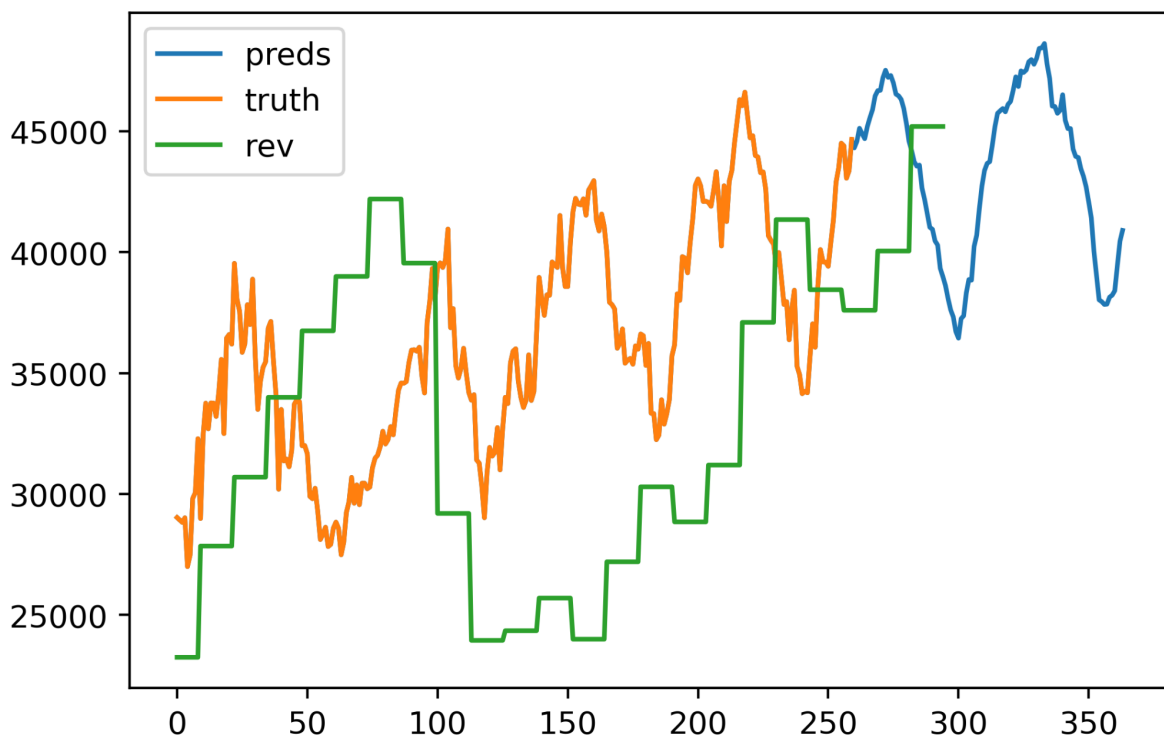




## Appendix B:

We initially wanted to use the trend of revenue and professional financial analysts' predictions of it into the future to aid in predicting demand for product A, however, we encountered many interesting observations that eventually rendered the revenue to be not valuable for predicting trends. We then turned our target into guessing the real-world identity of product A which is not that relevant to the challenge. Nonetheless, we still want to document our exploration below.

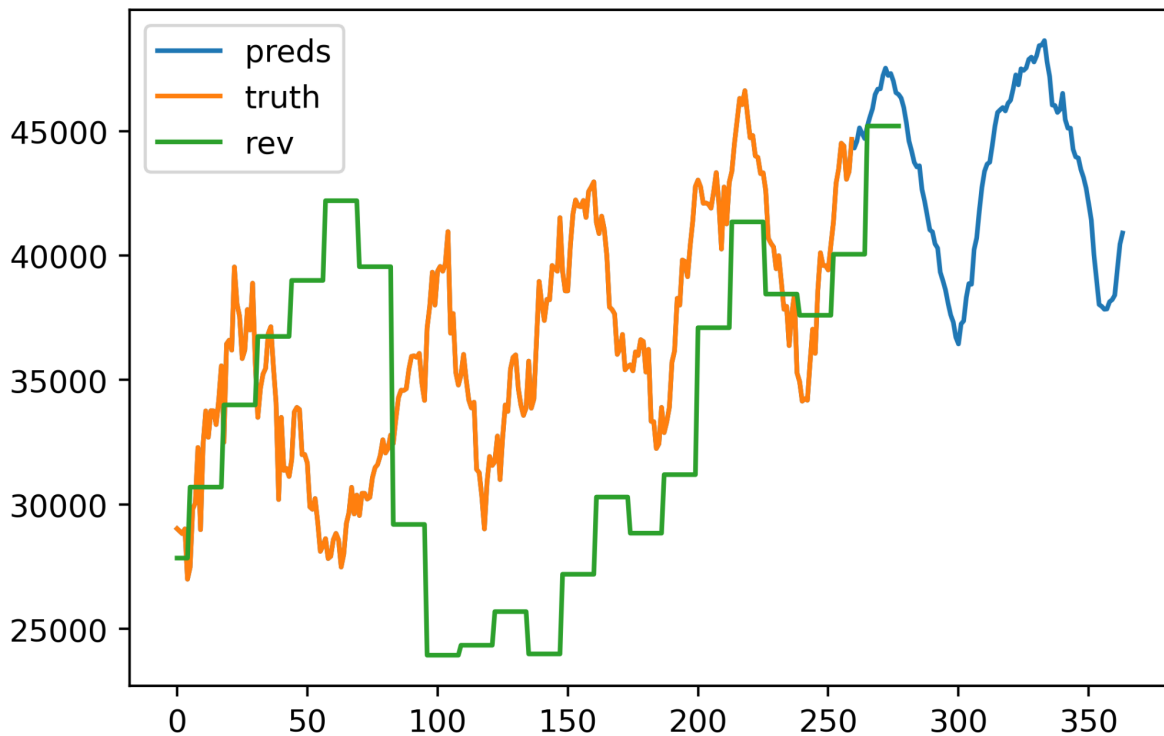
We obtained the past revenue per quarter of Micron from FY2017Q2 to FY2022Q1 (corresponding to 1st Dec 2016 to 30th Nov 2021), as well as the analysts' consensus estimate<sup>22</sup> for the coming quarters, and mapped them into weeks by taking each quarter to be 13 weeks. We then removed the first 4 weeks of data since Dec 2016 is not within our range of dates. For ease of visualization, we multiplied each quarter's revenue in billions by 5000 to obtain a similar order of magnitude as the demand data. The results, however, were perplexing at first glance:



While there is a certain degree of correlation between revenue and demand, they seem to be lagging a lot. We hypothesise that the time gap is due to the difference in accounting at

<sup>22</sup> <https://www.marketbeat.com/stocks/NASDAQ/MU/earnings/>

Micron - the time gap between the order was recorded and the time at which Micron was paid fully for the order. To investigate, we removed the first 17 weeks of revenue data, effectively shifting the revenue curve forward by 17 weeks. We were able to obtain a more correlated chart:



This tells us that the time gap is about 17 weeks, which for simplicity sake, we assumed to be the lead time of product A. Given in the case question extract that the average lead time during the semiconductor shortage in 2020 was 21.9 weeks, it is reasonable to estimate that a similar shortage but less severe in 2017 could result in the lead time of around 19 weeks, which is reasonably close to our observation from the graph.

However, there is still significant discrepancy between the demand and revenue around weeks 40 to 100, and the revenue during weeks 100 to 200 seems too low. We hypothesise that this is due to the fluctuations in DRAM and NAND prices during that period, which corresponds to Oct 2017 to end Nov 2018 and Dec 2018 to end Oct 2020 respectively. Because we dropped the first 21 weeks of revenue data, effectively, these periods correspond to earnings about 7 months later, which is May 2018 to end Jun 2019 and Jul 2019 to Apr 2020.

We perform the analysis below on the assumption that the demand of product A is recorded when a customer makes the order to buy it but the payment only comes after receiving it. We also assumed that the price of goods are agreed on when the customer orders it, not when he receives it. We also assumed that product A is a product that has enough revenue on its own to impact the company's total revenue meaningfully.

For weeks 40 to 60 (Oct 2017 to end Nov 2018), the prices for DRAM and NAND chips are constantly dropping while revenue kept increasing. From the earnings report of Micron during that period (FY2018Q2)<sup>23</sup>, we hypothesise that the demand for flash memory increased due to the industry-wide shift to cloud computing, outweighing the drop in prices. This contradicts with the demand data that we have. Due to lack of information and time to research, we are unable to arrive at a concrete explanation for the decrease in demand during that period, besides it being a seasonal trend. We think that the demand data is being intentionally simplified to remove a lot of non-seasonal trends.

For weeks 60 to 100, the revenue dropped significantly opposing the trend of demand increase. As mentioned by Micron's earnings report for that period (FY2019Q1 to FY2019Q3),<sup>24</sup> market environment and supply-demand dynamics were 'challenging', possibly resulting in a drop in the average selling price (ASP) of Micron's products. This could be caused by the rising inventory levels of hyper-scale cloud providers in preparation for Intel's launch of its next-generation server CPU 'Cascade Lake' in 2019 Q2.<sup>25</sup> Moreover, there was a shortage of low-end Intel CPUs at that period, which limited demand from PC and OEM side.<sup>26</sup> The excess of leftover mining GPUs from the burst of Bitcoin bubble in 2018 significantly reduced demand for GPUs as seen in Nvidia's earnings report, thus graphic memory products too. Combined, this resulted in a large drop in demand and thus ASP. Despite the conditions, product A still experienced its cyclic demand increase, albeit noticeably more subtle compared to other years. From here, we can guess that the demand for product A is only correlated to the greater market of DRAM and NAND to a small extent. With the possibility of product A being a PC/server DRAM product and high bandwidth product (graphic memory) being excluded, and NAND is also struck out due to its

<sup>23</sup><https://investors.micron.com/news-releases/news-release-details/micron-technology-inc-reports-results-second-quarter-fiscal-2018>

<sup>24</sup><https://investors.micron.com/news-releases/news-release-details/micron-technology-inc-reports-results-second-quarter-fiscal-2019>

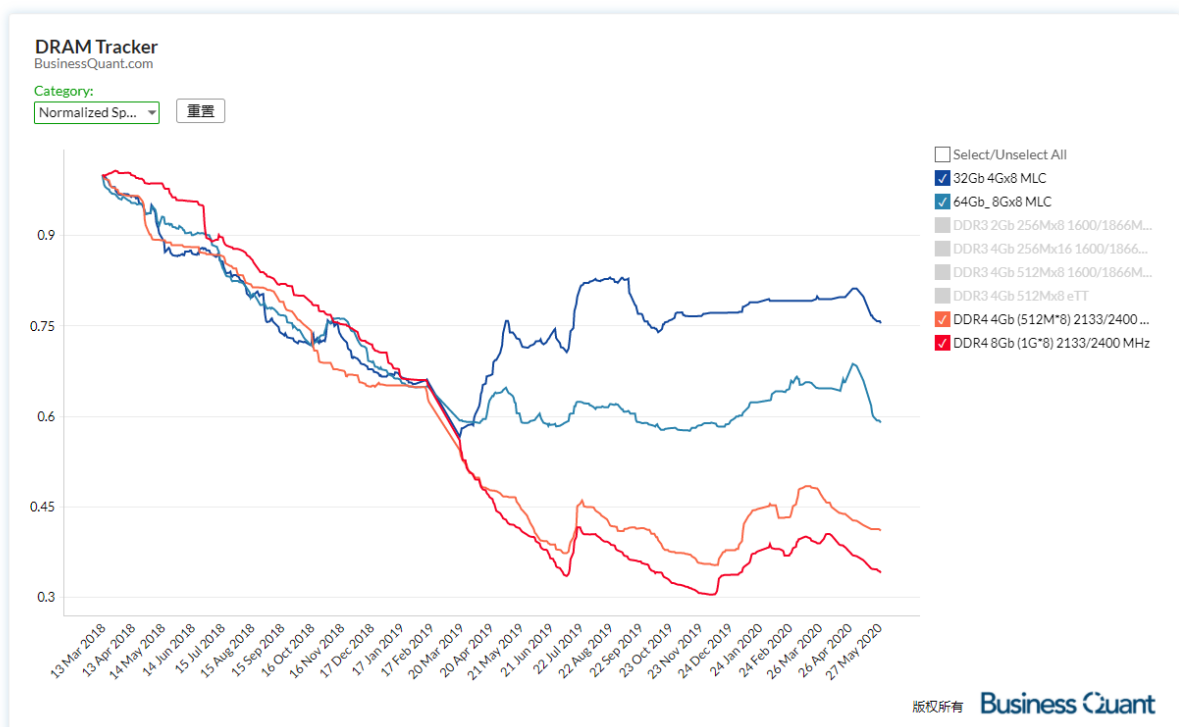
<sup>25</sup> [https://en.wikipedia.org/wiki/Cascade\\_Lake\\_\(microarchitecture\)](https://en.wikipedia.org/wiki/Cascade_Lake_(microarchitecture))

<sup>26</sup> <https://epsnews.com/2019/03/14/dram-prices-drop-30/>

non-seasonal nature, one possibility here is that product A represents a Low-power DRAM (LPDRAM) product for mobile devices, which is highly seasonal as the release of new generation of Qualcomm mobile phone chips are often in Q4 of the year, and thus phone manufacturers will start ordering the parts earlier in Q3. Mobile LPDRAM products are also not likely to be affected as much by the demand in server and PC markets, since the LPDRAM chips are rarely used on server or PC. Of course, a larger possibility would still be that the data provided to us was intentionally normalized to have a clear seasonal trend (as stated in the competition FAQ after we did this exploration).

For weeks 100 to 200 (Jul 2019 to Apr 2020), the revenue stayed relatively low while the demand increased. While the gap between them is constant, the correlation becomes much more apparent and sensible. The constantly underwhelming revenue is likely to be caused by weakening DRAM prices as competition intensifies in the industry and production costs reduces.

### Daily DRAM and NAND Spot Price Chart



Data from Business Quant<sup>27</sup> reflected a large drop in DRAM prices from Feb 2019 to Jul 2019, while the prices for NAND maintained a high level. As we see that the demand for

<sup>27</sup> <https://businessquant.com/dram-tracker>

product A continued to increase but the revenue stayed low, we can guess that the price of product A is related to DRAMs. Moreover, our analysis on the period of week 60 to 100 leads us to believe that product A represents a product for mobile devices, and only DRAM can allow for a discrepancy between PC product prices and mobile product prices, since the types of chips used are different, while for NAND chips, it is highly similar and interchangeable. Thus, our hypothesis that product A is a type of LPDRAM product for mobile devices can be further reinforced here. Some examples of such products would be the LPDDR4 and LPDDR5 chips.

From week 170 onwards, the correlation between revenue and demand become significantly stronger, leading us to believe that our initial assumption that the numbers given to us are representative of the seasonality seen in Micron's products but intentionally simplified to make forecasting easier for the competition is correct to a certain extent, despite the disclaimer stating that the numbers are fully arbitrary.

## Appendix C:

The second LP solution involves greater complexity. We assumed that machines only require maintenance after working the stated hours **consecutively**, meaning that they were able to work for a while, pause for 1 hour to reset the “cooldown”, and resume work. This allows for more productive efficiency.

The reason why this proof is more complicated is that we are not able to model the behaviour of individual groups of machines, which we previously could model using shifts. This makes it more difficult to verify whether a machine is working beyond its “cooldown” period.

Our proof for this relies on the assumption that the schedule must be the same for every single day. If we were able to change the schedule from day to day, our figures can be optimized even further; but this would require us to calculate a schedule of period  $\text{lcm}(23, 11, 10) = 2530$  days, and we are somewhat unwilling to do that much calculation.

In the following proof, we let  $x_i$  represent the number of Machine X's used in the  $i$ -th hour, instead of the number used in the  $i$ -th shift.  $X$  will then represent the total number of machines used, which is not necessarily  $x_1 + \dots + x_{24}$ . We also let  $I$  be the set of valid hours, i.e.  $[1, \dots, 24]$ .

The reason why we can no longer use the working shift notation is that we do not know **when** the breaks will be interspersed amongst the working hours to reset the “cooldown”. To model it using shifts requires us to list out all the optimal permutations which use  $k$  breaks as well as all the sub-optimal permutations which use  $k+1$  breaks, due to the possibility of non-integer solutions. For the rest of the variables, we use the same notation introduced in Question 3.

### Proposition 1:

$$x_{i \bmod 24} + \dots + x_{(C+i) \bmod 24} \leq (C-1)X.$$

### Proof:

We know that an hour-by-hour optimal solution will necessarily **not** include any situations where an individual machine works for their entire maximum consecutive period, as invoking the cooldown will necessarily require the machine to work for an equal or lower amount of time than if the cooldown reset was used. That is to say, each machine will work  $(C-1)$  consecutive hours at most to maximize efficiency. In order to verify that this condition holds in our machine distribution, we look at all 24 possible time windows of size  $C$ .

Summing  $x_i$  to  $x_{C+i}$ , we obtain the total machine-hours worked during that time window. By the pigeonhole principle, there must be at least one break during any such period of length  $C$  for the machine to not reach its consecutive working limit during any hour. Therefore, dividing this sum of machine-hours by  $(C-1)$  hours yields a lower bound for what  $X$ , the total number of machines, **has to be**.

**Proposition 2:**

$$x_i \leq X \quad \forall i \in I.$$

**Proof:**

It should be evident that you cannot have more machines of any type working in any hour than the maximum number of such machines used over 24 hours.

By taking all this into consideration, we are able to formulate the following:

Objective function:

- Minimize  $104X + 90Y + 82Z$  where  $X, Y, Z$  are non-negative integers.

Inequality constraints:

- $x_{i \bmod 24} + \dots + x_{(22+i) \bmod 24} - 21X \leq 0 \quad \forall i \in I.$
- $y_{i \bmod 24} + \dots + y_{(10+i) \bmod 24} - 9Y \leq 0 \quad \forall i \in I.$
- $z_{i \bmod 24} + \dots + z_{(9+i) \bmod 24} - 8Z \leq 0 \quad \forall i \in I.$
- $x_i - X \leq 0 \quad \forall i \in I.$
- $y_i - Y \leq 0 \quad \forall i \in I.$
- $z_i - Z \leq 0 \quad \forall i \in I.$

Equality constraints:

- $x_i + y_i + z_i = \text{number of machines required in hour } i \quad \forall i \in I.$

As such, our final answer in this case would be: **0**  $X$ , **7**  $Y$ , and **217**  $Z$  machines are required. The total cost of this on a daily basis is **\$1,842,400**.

The reason why the distribution is so skewed towards  $Z$  this time is due to the fact that **adjusting for cooldown resets on a day-to-day basis affects different machines to different extents**. Even with cooldown resetting,  $X$  can still only be used for 22 hours a day, whereas  $Y$  can be used for 21 hours a day (from the previous 20) and  $Z$  can also be used for

21 hours a day (from the previous 18). Therefore, Z instantly beats all the others in terms of cost efficiency and becomes prioritized by the algorithm. However, if schedules are not repeated daily but changed instead using the 2530-day rotation, we expect Y to remain the most used as Z's cooldown reset advantage under indivisibility will be removed.



## Appendix D:

Here we present the source code for solving the more complex linear optimization problem that we formulated in Appendix C. We used Google's open-source linear programming package [OR-Tools](#) which includes a very powerful **integer** linear program solver named [SCIP](#), something that is absent in some of the most well-known Python libraries like Scipy.

We used Python's dynamic arbitrary code execution (`exec()`) to our advantage to efficiently automate defining all the 75 variables and 168 constraints.

### Requirements:

1. Python 3.6+ on Linux, MacOS or Windows
2. Python package: OR-Tools by Google (install by running `pip install ortools`)

### Full code:

```
from ortools.linear_solver import pywraplp
solver = pywraplp.Solver.CreateSolver('SCIP')
infinity = solver.infinity()

# Variables
for i in range(1, 25, 1):
    exec(f'x{i} = solver.IntVar(0.0, infinity, "x{i}")',
globals())
    exec(f'y{i} = solver.IntVar(0.0, infinity, "y{i}")',
globals())
    exec(f'z{i} = solver.IntVar(0.0, infinity, "z{i}")',
globals())

X = solver.IntVar(0.0, infinity, "X")
Y = solver.IntVar(0.0, infinity, "Y")
Z = solver.IntVar(0.0, infinity, "Z")

print('Number of variables =', solver.NumVariables())
```

```

assert solver.NumVariables() == 75

# Constraints

#  $0x_1 + 1x_2 + 1x_3 + 1x_4 \dots -21X \leq 0$ 
#  $0y_1 + 0y_2 + 0y_3 + 0y_4 + 0y_5 \dots -9Y \leq 0$ 
#  $0z_1 + 0z_2 + 0z_3 + 0z_4 + 0z_5 + 0z_6 + 1z_7 \dots -8Z \leq 0$ 
# Cycle above 24 times, each time move the 0s forward by 1, x:
1 0, y: 14 0, z: 15 0

x_no_0 = 1
y_no_0 = 14
z_no_0 = 15

for i in range(24):
    # Make cyclic zero masks
    multiply_x = [1] * 24
    multiply_y = [1] * 24
    multiply_z = [1] * 24
    if i + x_no_0 <= 24:
        multiply_x[i:i + x_no_0] = [0] * x_no_0
    else: # fill rest of list with 0 first then cycle back to
front
        multiply_x[i:] = [0] * (24 - i)
        multiply_x[:i + x_no_0 - 24] = [0] * (i + x_no_0 - 24)

    if i + y_no_0 <= 24:
        multiply_y[i:i + y_no_0] = [0] * y_no_0
    else:
        multiply_y[i:] = [0] * (24 - i)
        multiply_y[:i + y_no_0 - 24] = [0] * (i + y_no_0 - 24)

    if i + z_no_0 <= 24:
        multiply_z[i:i + z_no_0] = [0] * z_no_0

```

```

else:
    multiply_z[i:] = [0] * (24 - i)
    multiply_z[:i + z_no_0 - 24] = [0] * (i + z_no_0 - 24)

# Add the constraints
# Total constraints for this: 24*3 = 72
x_sum_expr_code = [f'x{j+1} * multiply_x[{j}]' for j in
range(24)]
    exec(f'solver.Add(solver.Sum(["", ".join(x_sum_expr_code)"]))
<= 21*X)', globals())
y_sum_expr_code = [f'y{j+1} * multiply_y[{j}]' for j in
range(24)]
    exec(f'solver.Add(solver.Sum(["", ".join(y_sum_expr_code)"]))
<= 9*Y)', globals())
z_sum_expr_code = [f'z{j+1} * multiply_z[{j}]' for j in
range(24)]
    exec(f'solver.Add(solver.Sum(["", ".join(z_sum_expr_code)"]))
<= 8*Z)', globals())

# xi <= X for all x
# yi <= Y for all y
# zi <= Z for all z
# Total constraints for this: 24*3 = 72
for i in range(1, 25, 1):
    exec(f'solver.Add(x{i} <= X)', globals())
    exec(f'solver.Add(y{i} <= Y)', globals())
    exec(f'solver.Add(z{i} <= Z)', globals())

# x1+y1+z1 = K, for var1...24
# Total constraints for this: 24
K =
[172,205,201,190,170,205,173,179,176,200,199,199,190,208,194,1
84,208,209,195,193,197,207,209,188] # Obtained from Excel
provided

```

```

i = 0
for k in K:
    i += 1
    exec(f'solver.Add(solver.Sum([x{i}, y{i}, z{i}]) == {k}))',
globals())

print('Number of constraints =', solver.NumConstraints())
assert solver.NumConstraints() == 72 * 2 + 24

solver.Minimize(10400 * X + 9000 * Y + 8200 * Z)
status = solver.Solve()

if status == pywraplp.Solver.OPTIMAL:
    print('Solution:')
    print('Objective value =',
round(solver.Objective().Value()))
    for i in range(1, 25, 1):
        exec(f'print("x{i} =", int(x{i}.solution_value()))',
globals())
        exec(f'print("y{i} =", int(y{i}.solution_value()))',
globals())
        exec(f'print("z{i} =", int(z{i}.solution_value()))',
globals())
    print('X =', int(X.solution_value()))
    print('Y =', int(Y.solution_value()))
    print('Z =', int(Z.solution_value()))
    print(f'Problem solved in {solver.wall_time()}
milliseconds')
else:
    print('The problem does not have an optimal solution.')

```

This fully reproducible code can also be found online in our GitHub Repo:

[https://github.com/aliencaocao/BACC-2022/blob/master/linear\\_optimization.ipynb](https://github.com/aliencaocao/BACC-2022/blob/master/linear_optimization.ipynb)