Lifelong Learning for Closed-Loop Soil Moisture Control

UNIVERSITY OF
LINCOLN

Student Name: Jack Foster

Supervisor: Prof. Simon Parsons

MSc Robotics and Autonomous Systems

University of Lincoln, UK

September 10, 2022

Contents

List of Figures

List of Tables

Abstract

As the agricultural industry comes under increasing strains due to population growth, climate change, and geo-political strife, there is a need to increase crop yield to ensure food security in the coming years. As crop yield is intrinsically tied to soil-moisture, this work proposes a lifelong learning-based precipitation prediction model, paired with a bespoke sensor-based weather station, to facilitate the optimisation of soil-moisture. The project concludes with a novel extension of the elastic weight consolidation algorithm through application of a variational autoencoder to estimate an intractable likelihood term.

1 Introduction

The food-supply chain is expected to be under growing strain over the next decades due to challenges such as worker shortages, climate change, and population growth (Ambler-Edwards et al., 2009). Therefore, improving the efficiency of agricultural processes is of utmost importance; as well as ensuring urban agriculture becomes more widespread to alleviate stress on the primary food sources. One key factor in improving crop yield is ensuring an appropriate soil-moisture level is maintained (Pierce and Nowak, 1999). If the soil-moisture level of a crop bed could be controlled and managed, then the yield would be significantly improved. While using soil-moisture sensors may help maintain the moisture level by quantitatively measuring the current moisture levels, they do not reason about potential future states. This is a significant shortcoming as, for example, if the soil is currently rather dry but torrential rain is imminent, it would not be wise to water crops. Using solely soil-moisture sensors is limited in that such reasoning cannot be extracted from the current soil state alone.

To address this shortcoming, a long-short term memory model was developed to predict whether precipitation is expected within the next 24 hours. The model trained on existing datasets of sensor data, but to test the efficacy of the model it was then deployed in a real-world scenario, predicting precipitation to inform watering regimes for radishes on a small urban potted test bed. A bespoke sensor-based weather station was designed and deployed to obtain the necessary atmospheric and soil-moisture data. Sensor-based data is preferable to alternatives such as weather forecasts or satellite imaging, as weather is heavily region-dependent and sensors concern themselves with the exact location of the crops, rather than the general area. Furthermore, sensor systems are more accessible than satellite imaging data which may be hard to acquire, as sensor systems may be purchased and deployed manually without relying on access to resources such as satellites.

As precipitation data is highly localised, it is likely that adapting to a new region will significantly impact the performance of the precipitation model. Therefore, lifelong learning was used to fine-tune the model, ensuring performance was maintained after moving from existing datasets to deployment in a different geographic area. Furthermore, soil-moisture sensor data was used as feedback to create a closed-loop control system, providing automatically generated ground-truth labels to the lifelong learning network to facilitate the incremental learning of region-specific data. Using the soil-moisture data as feedback also allows immeasurable information such as the water-retention properties of the soil or the evaporation rates of the region

to be indirectly encoded into the ground truth labels. While these aren't measured directly, the rate of change of soil-moisture is dependent upon these properties and thus they are implicitly encoded into the training regime.

The final contribution of this work is a novel extension to the elastic weight consolidation (EWC) algorithm, which uses a variational autoencoder to estimate the intractable likelihood term within EWC on a sample subset of size $n > 1$, rather than on a per-sample basis.

To critically evaluate the efficacy of the contributions made, this paper seeks to address the following research questions:

- Can a neural network model accurately predict future precipitation?
- Is the future soil-moisture state pertinent to optimising soil-moisture levels?
- Does lifelong learning alleviate issues around regional perturbations in atmospheric data?
- Does the closed-loop control of soil-moisture data improve the soil-moisture levels of a crop bed?
- Can the EWC algorithm be improved with subset likelihood estimation with a variational autoencoder?

This project explores a variety of subject areas, ranging from theoretical alterations to likelihood estimation to the design and deployment of a physical weather sensor system. The completion of all tasks led to the development of a bespoke system that solves an important real-world problem, as well as contributing a novel extension to the EWC algorithm. This report will cover all topics explored within the project, starting with a literature review, before discussing the methodology of the research, the design and development of the artefacts created, as well as quantitatively evaluating the performance of said artefacts. Finally, the report discusses the implications of the findings, suggests future work, and evaluates the overall management of the project.

2 Literature Review

2.1 Precision Agriculture

Precision agriculture is a means of improving the efficiency and yield of agricultural sites through the granular optimisation of agricultural processes using technology and data analysis (Duckett et al., 2018). This can take many forms, such as using site specific weed management and precision spraying to only apply harmful herbicides in locations where it is needed (López-granados, 2011), or through the monitoring of crops and their environment to inform the development of optimal growth strategies. For example, a more precise and controlled watering regime may be achieved through analysing sensor data and making decisions based on ground truth observations, rather than a 'best guess' strategy based on a farmer's expert knowledge (Pierce and Nowak, 1999). This is useful for very large farms where relying on human analysis is infeasible, or for urban agriculture where the human agent may not possess the necessary knowledge to inform such decision making.

An important motivation behind the development of precision agriculture has been the need to better control variance in spatio-temporal parameters (Blackmore, 2009). Specifically, this concerns parameters that significantly influence crop yield, but are also sensitive to changes in space or time, as it can be difficult to control such variables in unpredictable domains, but doing so will improve yield (Zhang et al., 2002). Therefore, developing robust, generalisable models that can predict future states of these variables is of great importance. Combined with systems to measure the current variable state, these models may be used to inform decision making to optimise processes such as fertilising or watering the soil (O'Neal et al., 2000).

Soil is one such variable, specifically its moisture content and its physico-chemical properties. The properties of the soil will vary based on where in the farm a crop is planted, as well as conditions on a particular day. Proper soil management is important not only to increase yield, but to protect the arability of the land. As the UK has a shallow fertile topsoil (Duckett et al., 2018), it is vital that sustainable agricultural practices are applied that preserve its integrity, ensuring the long-term health and usability of such land. Over-watering can cause topsoil to become over-saturated, and so proper watering practices may improve the health of farmland.

Another key parameter is the spatially-variable topology of the field (Zhang et al., 2002). Given that farms are heterogeneous (Blackmore, 1994), developing systems that can estimate future

states in a non-uniform manner, will enable different subsections of a farm to be managed according to that section's needs, rather than receiving treatment that is optimal *on average* for the farm as a whole, but which may be ineffective for specific areas.

2.2 Urban Agriculture

Urban Agriculture, the process of growing fruits and vegetables in cities (Mougeot et al., 2000), can increase food availability during times of natural or socio-economic uncertainty; and thus may play a crucial role in addressing food shortages due to population growth and climate change (Beddington, 2011; Zezza and Tasciotti, 2008; Barthel and Isendahl, 2013). There are 3 major barriers to entry for urban agriculture: space, time and knowledge. The work presented here may alleviate issues present as a result of a lack of time or knowledge. Having a system that can automate the decision making of whether to water may reduce the cognitive load of the user. Micro-farming machines, such as those produced by Farmbot Inc (Barthel et al., 2015), are autonomous small-scale farms capable of completing tasks such as seeding, fertilising and watering. While effective at reducing the time component of maintaining a farm, these systems lack the capacity for autonomous decision making and rely upon user's knowledge. Work is needed to develop bespoke and robust prediction methodologies to inform the policies of robotic agents.

2.3 Neural Architectures

2.3.1 Discriminative Models

Neural networks are a data-driven computation method comprised of a series of non-linear functions. This means no a priori knowledge of inter-variable relationships is needed to learn complex input-output mappings. Therefore, neural networks may learn to solve tasks where such mappings are unknown or have not been explicitly defined. Multi-layer perceptrons (MLPs) are a form of feed-forward neural network that utilise a series of non-linear activation functions to map an input vector to a target output. These activation functions form layers of n computation nodes, each node in a layer passes its output to all nodes in the subsequent layer via a set of weighted connections. Figure 1 illustrates a generic MLP architecture. For the vast majority of cases, these weights are tuned through the back propagation algorithm (Rumelhart et al., 1985), which takes the updates weight values based on the partial derivative of the distance between the network output and a ground-truth label. In other words, each weight is updated via gradient descent such that it minimises its contribution to the network loss.

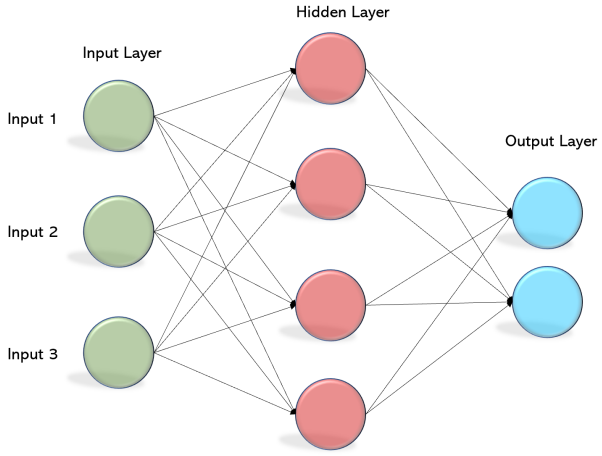

Figure 1: Multi-Layer Perceptron generic architecture

Unlike MLPs, convolutional neural networks (CNNs) utilise spatial information within the input vector when learning input-output mappings (Krizhevsky et al., 2012). This added spatial information is critical in many tasks, such as image classification or segmentation. The structure within the input vector may be to represent many things, not just spatial data. For example, the position of elements in a 1-D tensor may correspond to the position of words in a sentence, or the time at which a sensor reading was taken. This could be useful if the sensor readings are arranged with a temporal dimension, as the network could learn to utilise the change in sensor readings over a fixed sliding window of time. The ability to interpret this structural data is accomplished through the application of a convolution operation, which in practice is the element-wise multiplication of a kernel of weights with the input vector.

Another advantage of CNN architectures over MLPs is that as these kernels, or feature banks, are shared, CNN architectures are able to be substantially deeper, without the exponential rise in connections (LeCun et al., 2015). Increased depth facilitates the learning of more complex patterns, as early layers build generic feature extractors that identify rather low level features (in an image, they may extract lines or curves, for example) and later layers build on top of these to identify far more complex features (such as faces, or people).

While CNN models may be capable of extracting temporal information from the input vector, they do not retain or remember information. In contrast, long short-term memory networks (LSTM), proposed in (Hochreiter and Schmidhuber, 1997), are a recurrent neural network (RNN) model that are able to learn long-term dependencies. In other words, LSTMs are de-

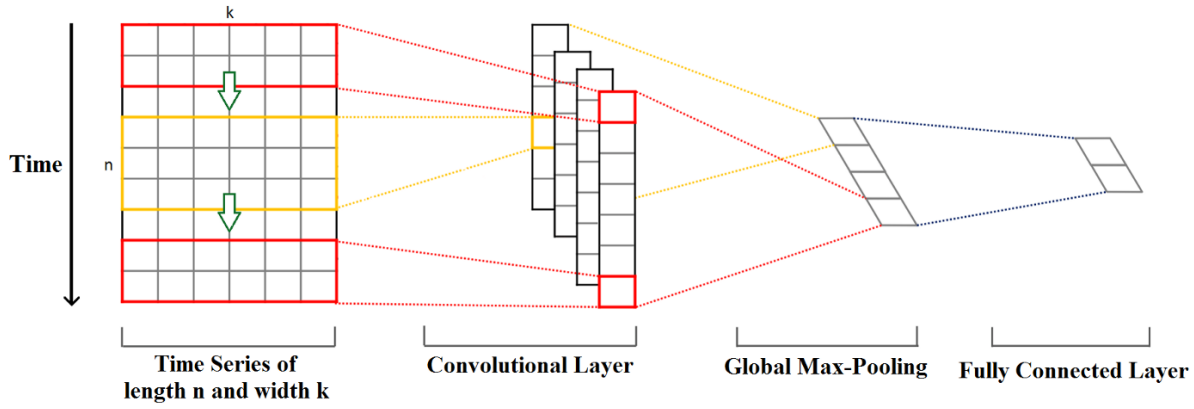

Figure 2: Time-series Prediction with CNN architecture

signed to remember information for long periods of time so that they may be used to predict later stages of a sequence. An LSTM node has four interacting layers, as seen in figure 3. The cell state, which passes information through the module with only a few linear transformations, is the primary means of long-term memory in the LSTM. Gates regulate cell state data flow, utilising a sigmoid layer to determine what percentage of a component should be passed through.

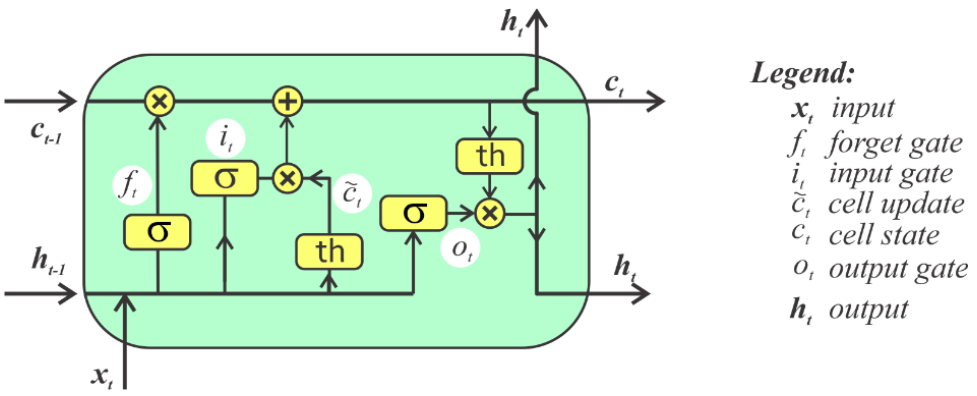

Figure 3: LSTM generic architecture

2.3.2 Variational Autoencoders

Autoencoders are comprised of an encoder and a decoder (Kingma and Welling, 2013). The encoder compresses an input vector (x) into a lower dimensional representation, known as the latent space (z). The decoder then reconstructs x from the latent space representation. The limitation of traditional autoencoders is that their latent space is sparse rather than a continuous distribution, and are therefore not generative models. If the model is to generate data other

than simply the reconstruction, the latent space must be continuous. To elaborate, if a sample is represented as a discrete point in the latent space, then the vast majority of the search space is empty and, when sampled, the decoder will not be able to decode it into meaningful information, resulting in noise. Variational autoencoders (VAE) solve this problem by generating a mean (μ) and a standard deviation (σ) for a given input sample, representing a normal distribution. The latent representation is then obtained by sampling from this distribution; the decoder then reconstructs the original input from the latent space. As gradient cannot be backpropagated through a random sampling node, the latent variables are separated from the sampling through the reparameterization trick (Kingma and Welling, 2013), allowing for the gradient to pass through the latent variables but not the random sampling. As z is obtained by sampling a probability density function, the same input sample can produce different latent representations, and thus the latent representation of a given input becomes a continuous distribution, rather than discrete. By combining the traditional autoencoder loss function with the Kullback-Leibler divergence (Kullback and Leibler, 1951), the clusters can be trained to fall within constrained bounds. The new loss function is defined as:

$$loss = \sum_{i=1}^{n} ||x_i - \hat{x_i}||^2 + \frac{1}{2}\sum_{i=1}^{n}(\sigma_i^2 + \mu_i^2 - log(\sigma_i) - 1) \tag{1}$$

$$loss = BCE(x, \hat{x}) + KLD(N(\mu_x, \sigma_x), N(0, 1)) \tag{2}$$

where x_i is an input vector and $\hat{x_i}$ is the reconstruction. Now that the latent space can be sampled, it is possible to generate new information from a given input as the distribution is not sparse. Finally, an important consideration of the VAE is that the encoder is approximating the distribution $P(z|x)$, and the decoder is reconstructing x from the latent representation z, which can be framed as maximising $P(x|z)$.

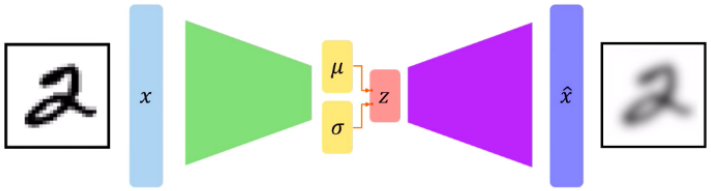

Figure 4: Variational autoencoder architecture (Amini and Soleimany, n.d.)

As the latent space of a VAE is a Gaussian distribution, and the network maximises $P(x|z)$, then under the right circumstances, the VAE may be used to extract terms for Bayesian estimation. Choi et al. (2020) estimate how difficult individual data samples are to classify during active learning using Bayesian estimation. One of the terms within the equation, the likelihood of a sample given a class label, is intractable. As such, it must be estimated using a variational autoencoder. This is achieved by applying a zero-valued mask to a subsection of the latent space for each class, representing a class as the absent of parts. The predicted label is then calculated as the lowest value in the latent space which is calculated using:

$$w = [\sum_{j \in C_1} z_j^2, \sum_{j \in C_2} z_j^2, ..., \sum_{j \in C_{N_c}} z_j^2] \tag{3}$$

$$\hat{y} =_n w \tag{4}$$

where z_j is the latent space for the j^{th} class. This constraint is enforced by adding a new loss function to the standard variational autoencoder loss function.

$$L_{class} = H(softmax(-w), \hat{y}) \tag{5}$$

Where H is the cross entropy and w is the transposed vector of the latent spaces denoted in equation 3. This can then be combined with equation 2 and a tuning hyperparameter, λ, to give the final loss function.

$$loss = BCE(x, \hat{x}) + KLD(N(\mu_x, \sigma_x), N(0, 1)) + \lambda L_{class} \tag{6}$$

As the latent space z is predicated on the class label ($class_j$), and the VAE maximises $P(x|z)$, the network can now be considered to be maximising $P(x|class_j)$, the intractable likelihood term. The ability to estimate intractable likelihood terms is useful in many applications, including in the lifelong learning methodology elastic weight consolidation (Kirkpatrick et al., 2017), covered in section 2.5.

2.4 Rain Prediction

Predicting precipitation is challenging as it concerns the close interaction of multiple volatile atmospheric processes; however, successfully predicting rainfall is key for many important applications, such as flood warning systems (Devi et al., 2016), or predicting future soil-moisture

levels. There are many works on rainfall prediction, and the majority utilise one of three forecasting models: auto-regressive moving-average (ARMA), neural networks, or the non-parametric nearest-neighbours (K-NN). ARMA models future rainfall as a linear function of historic data, and has been utilised to estimate short-term precipitation (Brath et al., 1988). However, this approach is limited in that it is linear, parametric, and model-driven (Toth et al., 2000). As such, ARMA models must first identify the type of inter-variable relationship, as well as an estimation of the parameters. This is limiting, and assumes an understanding of the underlying structure of the problem, thus it can only solve well-defined problems.

K-nearest neighbours, in contrast, does not require assumptions to be made about the inter-variable relationships, and is instead data-driven (Cover and Hart, 1967). As a non-parametric algorithm, K-NN models data-points based on their proximity to one another, as measured by a distance metric such as Euclidean distance.
K-nearest neighbour models are powerful and in addition to making few assumptions, possess an inherent adaptability when presented with additional data. However, K-NN models also have significant drawbacks. Firstly, the algorithm is slow, with a computational complexity of up to $O(ndK)$, where n is the cardinality of the dataset and d is the dimensionality of a sample (Veksler, 2004). Furthermore, setting the K hyper-parameter (i.e. the number of neighbours) is non-trivial and requires either a priori knowledge of the data, trial and error, or a Monte Carlo search.

The application of the MLP architecture to rainfall prediction is not novel, being first pioneered in (French et al., 1992). Here, an MLP with a single hidden layer was developed. The model predicted fluctuations in spatio-temporal rainfall intensity fields 1 hour in advance. The model was trained on historic data, and the work concluded that neural networks do have the capability to predict short-term perturbations in precipitation. This problem is considerably more trivial than predicting the precipitation for a full 24 hour period, and so it is likely an architecture with greater complexity will be necessary.

The hypothesis that neural models are well suited to precipitation prediction is further validated by Wu et al. (2010). The work achieved impressive results when a modular ANN was used in conjunction with singular spectrum analysis (SSA). SSA was used as a pre-processing step and attempts to decompose a time-series into the sum of a small number of explainable components, such as trends, periodic components, or noise (Hassani, 2007).

In Haidar and Verma (2018), 1-dimensional convolutions were applied to the temporal dimension of an input vector, building a set of temporally-aware features used to predict monthly rainfall. The mapping from input vector to output used in Haidar and Verma (2018) is given in equation 7. It shows how the time dimension of length m maps to m rainfall predictions. There are n variables for each time slice, which correspond to different atmospheric indicators which can be used to estimate rainfall. This method is effective due to the leveraging of the CNN's inherent ability to utilise the spatial information.

$$\begin{Bmatrix} \delta_1^1 & \delta_1^2 & \dots & \delta_1^n \\ \delta_2^1 & \delta_2^2 & \dots & \delta_2^n \\ \vdots & \vdots & & \vdots \\ \delta_m^1 & \delta_m^2 & \dots & \delta_m^n \end{Bmatrix} \rightarrow \begin{Bmatrix} r_1 \\ r_2 \\ \vdots \\ r_m \end{Bmatrix} \tag{7}$$

Many of the works reviewed rely on radar data or satellite imaging. Such methods, while effective, are not well suited to predicting the rainfall on a small or medium scale farm or for urban agriculture due to the inaccessibility of the data. Furthermore, given the large perturbations in weather with respect to geographic location, high geographic precision is needed to make accurate predictions about precipitation directly over the farming area if precipitation prediction models are to successfully inform watering regimens. In Qiu et al. (2017), atmospheric weather data is collected by a set of sensors located at several observation sites. This, combined with the 1D convolutions as outlined in (Haidar and Verma, 2018) and equation 7, could allow a CNN model to predict rainfall based on a time-series input vector with the n features relating to the set of sensor readings at each time slice. This is more desirable for the systems developed in this project, as sensor systems are more accessible than satellite data and offer better geographic precision. Furthermore, due to their small size, sensors lend themselves well to multi-agent problems. Multi-agent systems could be an appropriate solution to addressing the watering problem on large, heterogeneous farms.

As CNNs do not have a memory component, it can be challenging to predict precipitation multiple epochs in time and, therefore, recurrent networks may be better suited to precipitation forecasting. Luk et al. (2001) discusses how a wide range of atmospheric variables such as pressure or temperature may be mapped to predictions of the quantity and spatio-temporal

distribution of precipitation. As such processes cannot, in practice, be accurately modelled, non-linear mappings were learnt using several ANN models. The primary models used were: MLPs, partial recurrent neural networks, and time-delay neural networks. The study found similar performance across all models, thus highlighting that neural architectures are appropriate for precipitation prediction tasks. However, the recurrent model used here was an Elman network (Elman, 1990), which is largely inferior to the LSTM model and thus it is likely that an LSTM could surpass the performance of the tested networks.

LSTM models have already been applied to weather prediction problems (Poornima and Pushpalatha, 2019), (Qing and Niu, 2018). Poornima and Pushpalatha (2019) argue that traditional MLP models are not well suited to predicting rainfall since they do not take previous states into account (i.e. they are Markovian). To highlight the importance of historic data, an LSTM was developed to predict rainfall from atmospheric variables such as temperature, relative humidity, wind speed and hours of sun in the region. While successful, the study is limited in that all of the data was from the same region. Thus, to apply the model to a new region, it is necessary to first obtain a large amount of historic data for said region. This is impractical for scalablility purposes and for rapid deployment, and thus it is necessary to develop methods to transfer performance of a model to new regions.

2.5 Lifelong Learning

Lifelong learning is the process of continually acquiring, optimising and transferring knowledge over time (Parisi et al., 2019). Lifelong learning strategies enable models to learn solutions to new challenges while retaining solutions to previous tasks. This is useful for improving region-specific rain prediction as it could allow a model to learn to accommodate changes in the input-output mappings due to regional perturbations, without forgetting the original data. Retaining the original solutions is important as it prevents overfitting to short term weather data in the new environment. For example, if the model is deployed in summer it may quickly predict that there will be no rainfall, simply because rainfall is uncommon during summer. Therefore, when rainfall does eventually come, it will not be predicted. Furthermore, lifelong learning will allow the model to obtain additional implicit information from the soil-moisture sensor feedback.

Preventing catastrophic forgetting is of upmost importance in lifelong learning. Most deep neural models are trained on static datasets, if they continue to train once deployed, new information will interfere with previously learned knowledge (McCloskey and Cohen, 1989), leading

to catastrophic forgetting and consequently a massive drop in performance. As such, models must optimise across the antagonistic goals of learning solutions to new problems while retaining previously learnt behaviour.

Memory-based approaches attempt to mitigate catastrophic forgetting by retaining historic data, periodically reviewing it throughout a model's lifespan (Robins, 1993). The obvious constraint of this approach is that the data must be permanently stored, which may be problematic if the dataset is large or if there are hardware limitations. Memory approaches are inelegant and are undesirable for most tasks. Alternatively, additional neural resources may be allocated for new knowledge (Rusu et al., 2016). This methodology, however, may result in scalability challenges as architectures expand to accommodate new data. If the number of tasks is not known a priori then this approach becomes even more limited, as it is non-trivial to determine what resources to allocate before deployment.

Regularisation is a lifelong learning strategy that alleviates catastrophic forgetting through the application of penalties to the update of network weights (Parisi et al., 2019). This school of lifelong learning methods discourage the model from applying parameter updates that deviate the network weights from their original state, thus forcing the network to retain previously learnt information. Li and Hoiem (2017) introduces a regularization technique for convolutional models that utilises knowledge distillation. Here, knowledge is transferred to a small model from a large, regularised CNN, allowing information to be retained by compressing large amounts of computational information into a smaller parameter space.

Elastic weight consolidation (EWC), (Kirkpatrick et al., 2017), is a regularisation approach to lifelong learning. Instead of applying a network-wide quadratic penalty to learning, EWC instead assigns a penalty on a per-parameter basis, such that parameters that are more important to solving historic tasks are changed less than weights with less importance. This is effective as it maximises the computational capacity of the network, and encodes the network's past knowledge as a loss term. For the set of network parameters, θ, training a network maps the input to the output by finding the set of values for parameters, $\theta = \theta^*$, such that the loss is minimised. As it has been shown that the optimum network can be reached with many different configurations of θ^*, it may be deduced that there may exist parameter configurations that can yield near-optimum solutions for multiple tasks.

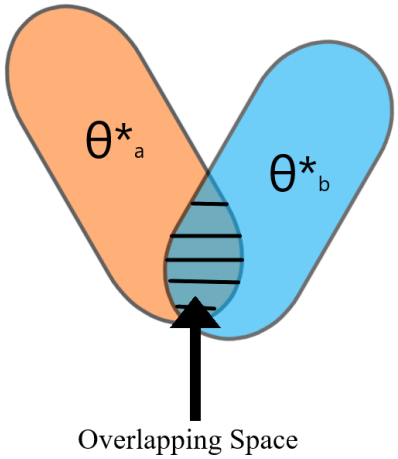

Figure 5: Optimal parameter configurations in parameter space, overlap can be seen between task A and task B parameter sets

EWC seeks to constrain network training to these overlapping sections, as seen in 5. This can be formulated as a Bayesian approach that estimates the network parameters θ, given data Σ as follows:

$$p(\theta|\Sigma) = \frac{p(\Sigma|\theta)p(\theta)}{p(\Sigma)} \tag{8}$$

As a logarithm function is monotonic, maximising a function is the same as maximising its logarithm, and thus $\log(\cdot)$ of 1 is taken:

$$log(p(\theta|\Sigma)) = log(p(\Sigma|\theta) + log(p(\theta)) - log(p(\Sigma)) \tag{9}$$

Taking a problem with two independent tasks, such that $\Sigma = \{A, B\}$, where B appears sequentially after A, then equation 9 may be written as;

$$log(p(\theta|\Sigma)) = log(p(B|\theta) + log(p(\theta|A)) - log(p(B)) \tag{10}$$

Here, $p(B|\theta)$ is in practice the loss for the current task, $p(B)$ is the likelihood for B, and the posterior probability $P(\theta|A)$ becomes the prior probability for task B. However, $p(\theta|A)$, which encodes information about the parameters acquired during the training for task A, is intractable. As it cannot be directly calculated, the posterior must be approximated. EWC solves this using a Laplace approximation. The term $log(p(\theta|A))$ represents the log-likelihood of the posterior probability density function $p(\theta|A)$, and as such is the inverse of the Fisher information matrix (FIM), which is the second derivative of the log-likelihood evaluated at the maximum likelihood estimate (Pawitan, 2001). As a network is considered to be trained when

the objective has reached a minima, the curvature of the loss surface represents the sensitivity of the network with respect to the optimum θ^*. Hence, the curvature is inversely proportional to the change in θ^* and thus a large curvature results in a large increase in loss. The curvature of a curve is represented by the second derivative, and thus the Fisher information matrix, F_A (which is comprised of second derivatives), can be used to estimate the sensitivity of a network with respect to θ^*, and thus the importance of each individual parameter within the network.

Once the posterior $p(\theta|A)$ has been approximated, the final value $p(\theta|\Sigma)$ can be calculated and used to inform weight updates. Following a series of simplifications, the final overall loss for the network using EWC then becomes:

$$l(\theta) = l_B(\theta) - \frac{\lambda}{2}(\theta - \theta^*_A)^T F_A(\theta - \theta^*_A) + \epsilon' \tag{11}$$

Here, $l_B(\theta)$ is the loss for B, F_A is the Fisher information matrix, ϵ' accounts for any constants, and finally λ is a hyper-parameter that allows the user to control the trade-off between retaining knowledge of task A and learning task B. It should be noted that the Laplace approximation is limited, as it assumes a leptokurtic distribution. That is, it assumes a smooth distribution with a large peak at its maximum point. Elastic weight consolidation has been shown to excel at learning new permutations of the same task Kirkpatrick et al. (2017). This is applicable to the problems addressed in this work, as learning to predict weather in a new location from sensor data is a permutation of the original task. As such, EWC is the most promising method surveyed.

3 Methodology

3.1 Programming Environment

Python3 was selected as the programming language of choice (Van Rossum and Drake, 2009), as the language contains the industry standard machine learning libraries Tensorflow (Abadi et al., 2015) and PyTorch (Paszke et al., 2019). Furthermore, Python3 contains the necessary web functionality to interface with the hardware weather station via a series of HTTP requests.

The choice between TensorFlow and PyTorch is fairly arbitrary. PyTorch was selected as the machine learning library as, despite its slower deployment time, it has a stricter object oriented paradigm - a feature which makes the design and maintenance of custom neural networks far easier to manage (especially for larger projects).

The Arduino programming language, built upon C/C++, was used to write the weather station code that runs on the microcontroller. This is because it contains many libraries custom written to facilitate interfacing with sensors and can abstract away the low level complexity of this process. As such, the Arduino language was by far the best choice in order to keep the code simple and useable.

3.2 Existing Data

Several public datasets are available that are designed to help develop precipitation prediction models. The MeteoNet dataset is the largest viable dataset identified, with 40GB of weather data for the north-west of France (Larvor et al., 2020). The data provided is in many different forms, but the most important data is a series of sensor readings for pressure, temperature, humidity, wind speed, wind direction, dew point and precipitation, taken from ground stations in the region every 6 minutes. This type of data is of particular interest as inexpensive sensors may be used to obtain similar data from an agricultural environment. This dataset was selected to train and test models against as it is sufficiently large, and cheap consumer hardware may obtain data in the same format (which is necessary if the prediction model is to be deployed).

The data available in the NCAR Rainfall Prediction dataset (Sharma, 2021) is of a similar format, containing daily atmospheric weather readings from sensor data taken from 1947 to 2018. This dataset was used to evaluate the efficacy of lifelong learning on new tasks, as its long date coverage and different geographic locations make it considerably different to the MeteoNet

data, despite having the same input data format, and thus is ideal as a permutation task dataset.

Once obtained, the data was parsed and normalized to make it appropriate for neural network input. Firstly, as the goal of the project was to predict the precipitation for the next 24 hours, a new column was added named 'daily precipitation'. This was simply the sum of the precipitation across all 6 minutely readings for a given day and ground station. Next, the data was collapsed into daily readings, by taking the first set of sensor sensor readings for each day - ground station combination, taken at midnight. Instead of having a sensor reading every 6 minutes, and a value for precipitation that occurred during that window, each day was instead represented by a single set of readings taken at midnight, with a column of the total precipitation for the next 24 hours. Many entries contained null values, where a specific sensor was not taking readings at that time. In such instances, all rows for that day and ground station were dropped, even if only a small number of the readings that day failed. This is because it would be otherwise impossible to get an accurate reading for daily precipitation.

The sensor readings were of wildly different scales. For example, the temperature taken in Kelvin was in the range of 250k-300k, whereas relative humidity is a percentage. Furthermore, wind speed was taken in ms^{-1} and barometric pressure was measured in Pascals, on the order of 10^5. As such, neural network behaviour would be heavily impacted by the metric with which each value was used. To alleviate this the dataset was normalized with the equation given in equation 12. The precipitation column was unchanged, as it was not used as input to the network but as a ground-truth label.

$$x_i = \frac{x_i - \mu}{\sigma} \tag{12}$$

Where x_i is a data point, μ and σ are the dataset mean and standard deviation.

3.3 Deployment

To truly evaluate the efficacy of the solutions proposed for soil-moisture optimisation, real world deployment was necessary. A flow chart of the system's decision making and order of execution can be seen in figure 6. Four pots of radishes were planted, and each pot was managed by a different watering regime. Radishes were selected as they can be planted up until October, and they grow within 21 days of being planted, making them ideal for rapid deployment and evaluation given the time constraints. The crops were watered at 12am every day if their decision making system dictated it. This was for two reasons: firstly, watering late is desirable as it

prevents the leaves from burning from the sun shining through water droplets and, secondly, the LSTM network was trained using sensor readings taken at 12am and thus it maintained consistency - atmospheric parameters experience substantial variation throughout the day, so it is desirable to standardise this. When the crops were watered, they received 300ml of water every time, to prevent extraneous variables influencing the result. 300ml is a large amount of water, but the pots had a large volume and so substantial watering was necessary.

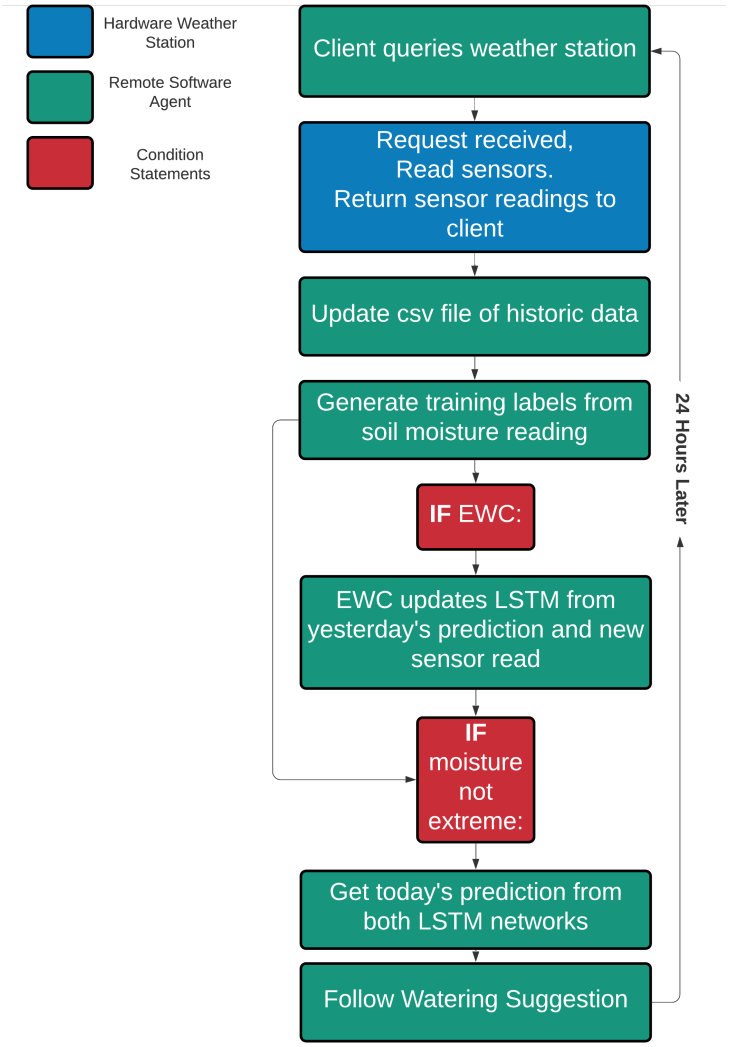

Figure 6: Step by step flowchart of system execution. Repeats at every day at midnight

The first pot was managed by a human agent. At the end of each day, they decided if they would like to water their pot based on their own observations of the soil and of the weather. These decisions were not based off of data or sensor readings, but rather more ”intuitive” notions such as sight or touch. This pot was designed to replicate how an average urban garden or small farm would be managed.

The second pot was managed by solely the soil moisture sensor. Soil moisture was measured in the range $[0, 1000]$. An optimal soil-moisture would yield a reading of around $550 - 600$. If the moisture level was below 550, the pot was watered, otherwise it was not. The binary split was slightly biased towards watering more often, as it is better for the radishes to receive too much water than not enough. However, the value of 550 was selected rather than 600, as there must be some allowance for the chance of rain (as there is no model prediction). If watering occurred up to a reading of 600, then days with rainfall would experience significant water-logging. This pot measured the importance of hard data when deciding the current state of the soil moisture. It was also used as a baseline to compare LSTM performance to, as this only considers the current state of the soil.

The third pot was managed by the base LSTM and the fourth by the EWC LSTM. For these pots, if the moisture was below a certain threshold (400), the pot would be watered regardless of the prediction. Similarly, if the moisture was above a certain threshold (700), it would not be watered. This is because the network's have no notion of the current moisture level and so while they may be predicting future state, if the current state is so extreme that the future state is irrelevant, then action must be taken irrespective of the prediction. For example, if the network predicts no rain but the soil is already extremely waterlogged, the soil should not then be watered just because it isn't going to rain that day. This has a larger allowance for dryer soil as if the network predicts it will rain, then the soil should soon be sufficiently watered for all but the absolute dryest of soil. In contrast, if the soil moisture is already very high, the absence of rainfall does not imply high levels of evaporation from the soil, and so less lenience was given in this regard. Finally, as the LSTM models took readings from the last 7 days as input, the atmospheric values were recorded for the week leading up to deployment, to allow the model to be used immediately.

The threshold moisture values used to determine when to water as well as the target moisture were obtained via guidance from the soil moisture sensor's documentation, which indicated what values would constitute dry, moist or waterlogged soil. The exact optimal soil moisture will vary depending on crop; radishes require a rather large amount of water and so an optimal value slightly skewed towards being waterlogged (but is still very much in the 'moist' category). If the target moisture is wrong, then the radishes may not grow correctly. However, comparison of the methods will still be valid as their ability to maintain a desired moisture level can still be

evaluated, and this target can simply be changed in future work if a better value becomes known.

3.4 Sensors

Once the radishes are planted, soil-moisture and atmospheric data must be collected from the planting area at set times, to allow the software agents to make watering choices, and to allow the neural networks to predict precipitation.

The ESP32 microcontroller is an Arduino-like device that can be programmed from within the Arduino environment and can interface with the same hardware sensors. The two primary advantages of the ESP32, however, are that it is about as small as an Arduino Nano, but it comes with inbuilt WiFi capability. This allows the device to communicate with external GPU-enabled devices such as a Desktop PC or cloud instance, thus allowing much of the complex computation to be done away from the farm environment. Equally important, it allows data to be extracted regularly to prevent any catastrophic loss of data.

The moisture content of the soil must be measured to assess whether the crop-bed requires watering. Capacitive soil-moisture sensors pass an electric current across two probes, measuring the resistance. As water has a lower resistivity than soil, the resistance readings are inversely proportional to the soil-moisture content (Eller and Denoth, 1996). The DFROBOT SEN0114 sensor was chosen as it was the cheapest Arduino-compatible soil-moisture sensor. A disadvantage of this sensor is it is not impervious to the elements, nor to the soil conditions. Over time these sensors will corrode and break, unlike alternative sensors available. However, these waterproof and corrosion resistant sensors such as the DFROBOT SEN0308 are considerably more expensive (approximately twice as expensive as the SEN0114). While these better soil moisture sensors may be preferable, financial limitations of this project dictated that the more affordable model be selected. As discussed in section 4.3, steps were taken to minimise the impact of these downsides.

Similar to the moisture sensors, the atmospheric sensors needed to be affordable, effective, and Arduino compatible in order to be considered a viable option. To calculate the pressure data, the Adafruit BMP280 is an effective consumer-grade barometric pressure sensor that is a simple i2c connected device that is small and very cheap. An added benefit is that the BMP280 is popular and thus well documented, so prior work exists to aid development time. The DHT11

sensor is designed to take readings of both humidity and air temperature, which are key atmospheric variables in the domain of rainfall prediction. Small and cheap, the DHT11 sensor is especially useful as it can collect two atmospheric variables with a single device. For all of these sensors accuracy is always of some concern given their cost, however the sensors selected were deemed to be the most appropriate within the financial constraints of the project.

4 Design and Development

4.1 Neural Network Development

4.1.1 Regression

From the surveyed literature, three types of neural network were dominant in the field of precipitation regression: the multi-layer perceptron (MLP), convolutional neural network (CNN), and the recurrent neural networks. As such, an MLP, CNN and LSTM were all tested, aswell as a convolutional LSTM. All networks took as input humidity, temperature, dew point, pressure, windspeed and wind direction. Dew point is the temperature the air needs to be cooled to (at constant pressure) to reach 100% relative humidity.

The MLP contained 2 hidden layers, of width 24 and 48 respectively. The output was a single node that output the amount of rainfall. Stochastic gradient descent was the optimiser used.

The CNN contained four convolutional layers with output channels of 12,18,24,30, respectively. Kernel size ranged from size 1 to size 6, and stride was kept at size 1. These convolutional layers were followed by two fully connected layers, eventually resulting in a single node output.

Finally, several LSTM architectures were tested, as they were intuitively the most appropriate model. The base LSTM model was 2 layers, with 12 hidden features, and was not bidirectional. The output of the LSTM was fed into a fully connected layer to output a single value. Built upon this core architecture, one model contained an additional fully connected layer between the LSTM and output layer. The intuition behind this added layer is that while the LSTM may be good at predicting future states, the input is a set of atmospheric variables, not the precipitation itself. As such, it may be beneficial to have an added layer to help with the translation from the future states into an actual precipitation amount. The second adaptation was a convolutional LSTM. Here, the input variables were first convolved into features, before these features were propagated through the LSTM and into the output layer. The motivation behind this design was that the convolutional layers and the LSTM layers have different strengths and weaknesses. If the convolutional layers are extracting important features using their spatially sensitive nature, it may be important to first obtain these features before the LSTM processes the features, rather than the raw input.

Rectified linear activation units were used for all models, as they are simple, fast to calculate,

and do not suffer from vanishing gradients as much as alternative functions. Root mean square error loss (equation 13) and root mean square error were used to calculate the difference between the predicted rainfall and the ground truth precipitation. MSE is the standard choice of loss function for regression tasks, but the loss is on a different order of magnitude to the prediction, and so it can be hard to introspect on model performance or to infer how well the model is doing with the given data. Taking the square root of the loss (RMSE) helps alleviate this problem. Finally, for all CNN and LSTM models, the Adam optimiser was used, as it has been shown to converge faster than stochastic gradient descent.

$$L = \sqrt{\frac{1}{N}[\Sigma(\bar{X} - X)^2]} \tag{13}$$

Unfortunately, the results across all models were terrible for regression. Further introspection into the problem indicated two primary issues. Firstly, the problem of predicting precipitation for the next 24 hours from a single reading is non-trivial. This is even harder when framed as a regression task. Secondly, the data distribution was such that the vast majority of precipitation fell below 5mm (figure 7).

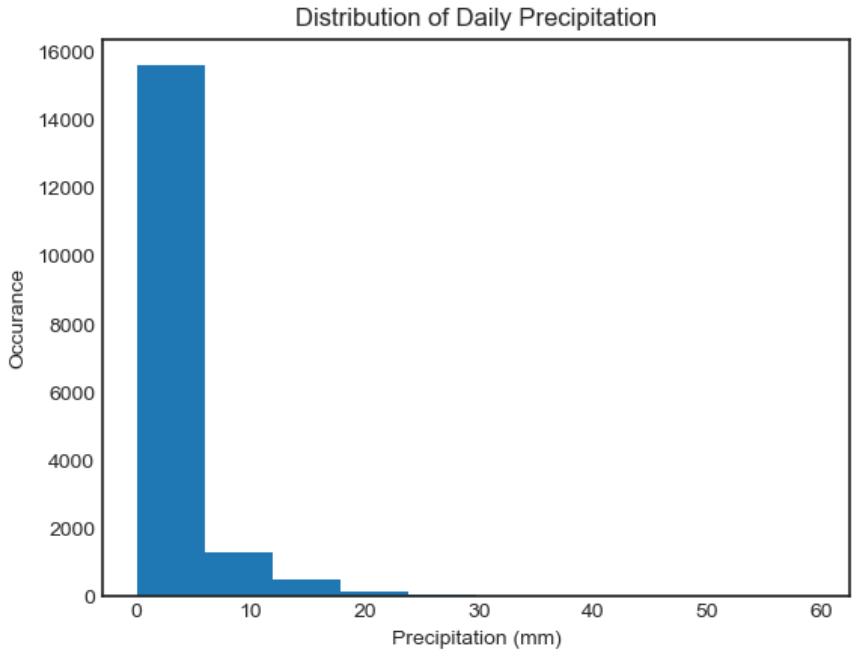

Figure 7: Distribution of Daily Precipitation in the MeteoNet Dataset

As such, it becomes very difficult for the network to predict anything above the 5mm threshold due to a lack of representation. Furthermore, the difference between rainfall per sample is in general far more granular than the range of the overall dataset; that is, because most of

the dataset is condensed into a small subsection of the actual data range, either the rest of the dataset is ignored or the model cannot accurately discriminate between samples that lie between 0-5mm. The real-world application of the network is to inform a watering regime, and this is largely a binary problem: if it rains significantly the crop should not be watered, otherwise it should be. As such, it was decided that moving to a classification approach would be acceptable, and to classify the dataset between significant precipitation or not.

4.1.2 Classification

The models were kept the same for the classification task, with a few notable alterations. Instead of a single node, the output was instead changed to a two node output representing the two classes "precipitation" and "no significant precipitation". The output of the model was given to a cross entropy loss function. In PyTorch, the cross entropy loss function first uses a softmax layer, converting the network output into a set of probabilities (eq. 14), hence the lack of softmax layer in the network itself.

$$\sigma(X)_i = \frac{e^{x_i}}{\Sigma_{j=1}^{K} e^{x_j}} \tag{14}$$

The UK met office defines light rainfall as anything below 2.5mm, and so this was taken as the threshold for the classes (i.e. anything above 2.5mm would be considered significant rainfall that would prohibit watering of the crop). While a third class representing moderate rain was initially used, it reduced overall performance and was not a necessity for the project, because radishes need 3.62mm of rainfall a day, and so anything above the 2.5mm rainfall would preclude watering the crops anyway. Thus, the problem was framed as a binary classification problem.

The models performed better in a binary class setting, but further introspection into the results indicated that there was a massive class imbalance ($\sim$ 3.5 times more samples in the majority class) and so the networks were simply predicting the same class repeatedly.

To alleviate this each class was assigned a loss function weight, such that the model was penalised more for incorrectly classifying a minority class sample. This helps even out the training of the network, as it offsets the bias of over-representation of the majority class. With this change, model performance improved, but one more change was tried to reach even better results.

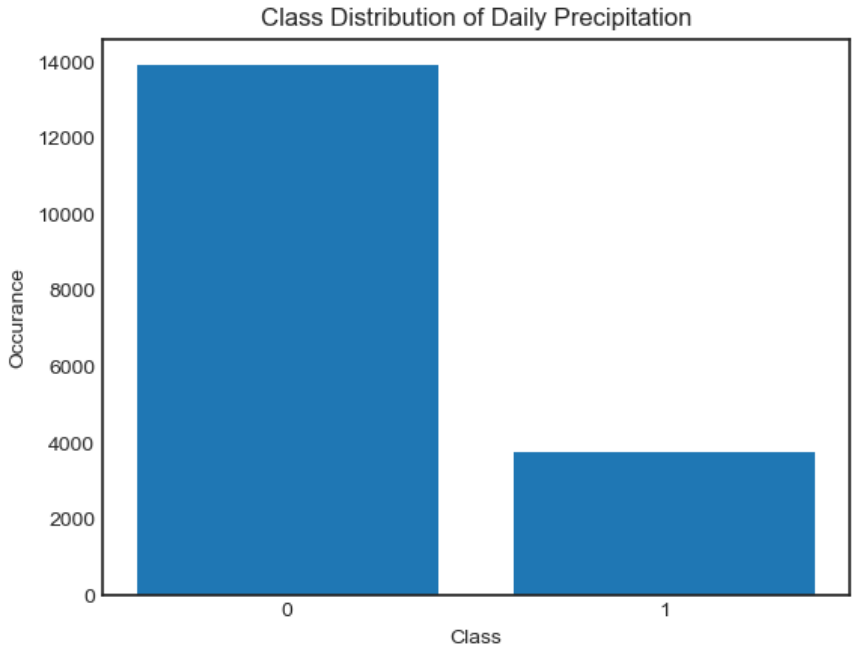

Figure 8: Class Distribution of Daily Precipitation in the MeteoNet Dataset

It was desirable to take a Markovian view of the prediction, whereby only the current set of sensor readings hold importance for the prediction and not past states, as this removes the need for any setup time and would allow the model to be deployed anywhere and make a prediction immediately based on one set of sensor readings. However, to evaluate the importance of historic data on the prediction, a time-series approach was also tested. Here, a CNN and a LSTM were given a set of sensor readings every day for the past week, with the task of classifying the next 24 hours into one of the two classes. This led to improved performance from the LSTM with two feedforward layers appended to it, surpassing the other methods. Therefore the LSTM model was taken forwards as the model to be deployed. An added advantage of the LSTM is that the ability to better learn long-term dependencies, unlike MLP, CNN and even other RNN models, offers significant power for the lifelong learning task explored in this work.

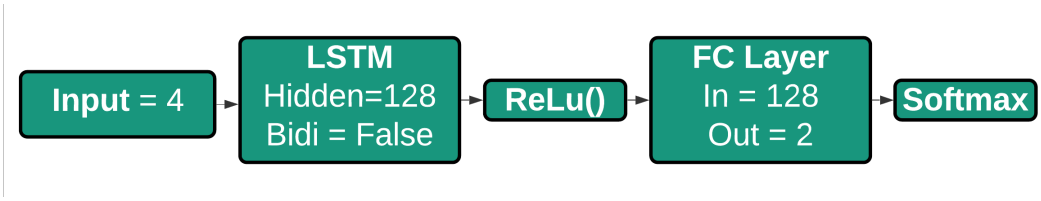

Figure 9: Final LSTM precipitation classification model

A constrained grid search was used to optimise the model's hyper-parameters. One interesting note discovery from this grid search, was that the windspeed and direction did not significantly improve the network performance. These variables were therefore removed from the network input, removing the need for wind state sensors such as anenometers. This was desirable, as these were the largest and most expensive of the sensors and so removing them allows the system to be lighter, more compact and far cheaper; all desirable traits for a device that is targeted at both commercial and consumer farms, and that may be part of a multi-agent system in the future. As such, the input vector was reduced to just the pressure, humidity, temperature and dew point across the last 7 days. An additional advantage to using historic data is that it may be particularly useful when paired with the soil-moisture sensor feedback with lifelong learning, as discussed later in section 4.2.3.

4.2 Lifelong Learning

4.2.1 Permuted MNIST Task

With an effective classification network developed, lifelong learning was implemented using elastic weight consolidation (EWC). EWC was chosen as it is particularly effective at permutation tasks, which is pertinent here. The initial implementation of EWC was adapted from the open-source github repository available from Hataya (2018). The initial testing of EWC was done using a simple feedforward MLP on the 3 tasks: the base MNIST task (LeCun et al., 1998), and two different MNIST permutation tasks (Goodfellow et al., 2013). The dataset is permuted in N different ways (i.e. pixels shuffled) to represent N new tasks. As the pixels are shuffled in the same way for every sample in a given task, there is still structure to be learnt to successfully classify. The permutations have been shown to be sufficiently different as to constitute distinct problems for a network (Farquhar and Gal, 2019). An MLP was used to maintain consistency between this experiment and the one in the original paper (Kirkpatrick et al., 2017).

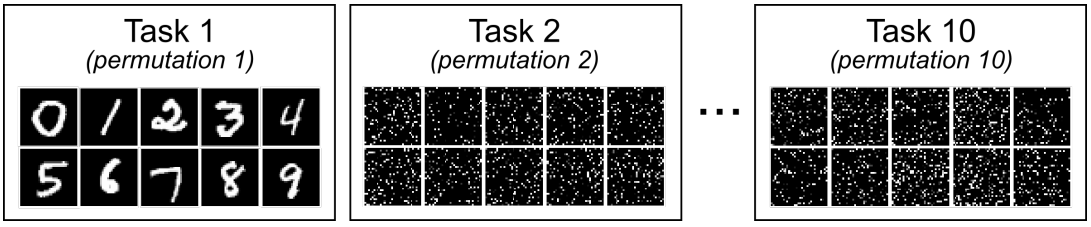

Figure 10: Permuted MNIST Tasks, source: `https://www.arxiv-vanity.com/papers/1904.07734/`

4.2.2 EWC for Precipitation Classification

Once EWC was shown to be working, it was implemented with the precipitation LSTM and the weather datasets. The first task presented to the network was to classify the MeteoNET data based on future precipitation (i.e. the same task as when developing the LSTM). The next task was to predict the wind direction for the same dataset. If the wind direction was less than 180 degrees it was set as label 0, and if it was more than 180, the label was set as 1. The final task was to classify the precipitation from the NCAR dataset. As the class distribution was different across all 3 tasks, a set of class loss weights were added to the loss function, differing for each task. For task 1, the weights were $[0.3, 1]$, for task 2 they were $[1, 0.95]$ and finally for task 3 they were $[1, 0.15]$. During training of the LSTM, the EWC network would randomly sample 200 datapoints across previous tasks and pass them through the LSTM, using the resultant gradient generated for the weights to calculate the importance of each weight. Note that the gradient isn't actually used to update the weights, they are calculated to calculate the EWC probabilities but then they are discarded. Once the EWC loss was calculated, it was added to the loss of network such that:

$$\mathcal{L}_{total} = \mathcal{L}_{network} + \lambda \mathcal{L}_{ewc} \tag{15}$$

where λ is the importance parameter.

A large challenge was setting the importance hyper-parameter, which modulates the magnitude of the penalty applied to the weights. Once an appropriate importance was found, the LSTM was able to perform well on all tasks while retaining knowledge of the previous ones.

4.2.3 Closed-Loop LSTM Elastic Weight Consolidation

For the precipitation problem, training on a dataset from France and then deploying in the UK is actually more of a transfer or continual learning challenge, not a lifelong learning challenge as the task is the same. However, there are several issues that are not addressed by these alternate strategies. Firstly, the original dataset has samples for all seasons, but deploying a model means that it has to experience each season sequentially in real time. This would lead to the model overfitting to the current weather and would not be able to perform in different seasons, or potentially even within the same season if the weather changes substantially. By retaining knowledge from the initial task, the ability to generalise to new seasons while still learning region-specific differences should be maintained. Another problem is that the network needs feedback to train. Obtaining the ground truth label of whether it rained substantially or not is non-trivial, and would likely require the human to manually insert this data. Finally, the

network is being used to inform watering regimens; while future rainfall is the largest factor in determining the soil moisture, there are many intangible variables that also impact the moisture, such as the water retention properties of the soil or the rate of evaporation. By framing the lifelong learning approach as a closed loop control system, the soil moisture sensor values taken 24 hours after the initial prediction can be used as feedback into the network. This is accomplished by binarising the soil moisture values, thus transforming them into a ground truth label. This process can be seen in the flowchart in figure 11.

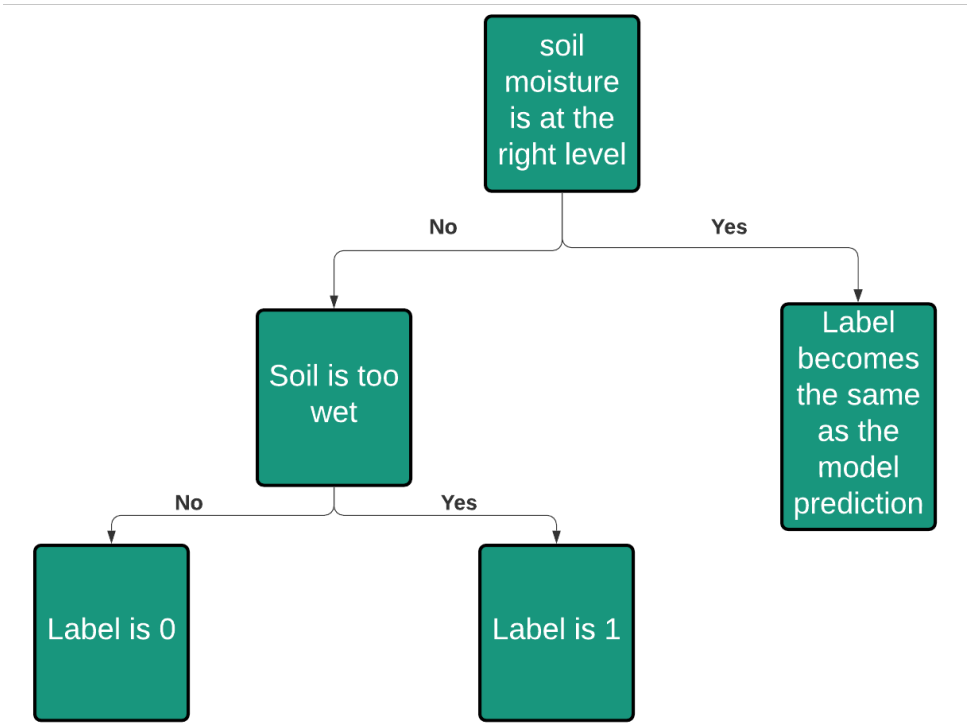

Figure 11: Soil Moisture to Ground Truth Label Flowchart

To elaborate, in the original network the label 0 corresponds to no rainfall expected (ergo the crop should be watered) and label 1 corresponds to rainfall is expected (thus the crop should not be watered). Here, the soil moisture is converted to label 0 if the soil is dry and 1 if it is too wet. This way, the labels are still semantically related to the rainfall prediction, as most of the time the presence or absence of precipitation will be the primary driving factor behind whether or not watering should occur, but by using the soil moisture values as labels other properties of the soil and atmosphere may be indirectly encoded. If the soil moisture values fall within the range deemed acceptable, then the ground truth label is set equal to the network's prediction, as it was the correct decision.

The final LSTM EWC network was deployed, using the same base LSTM trained on the MeteoNet dataset, with an importance of 250 and the soil moisture sensor feedback integrated to close the loop and facilitate autonomous training.

4.3 Weather Station

4.3.1 Circuitry

Initially, the weather station was to contain the DHT11 humidity and temperature sensor, the BMP280 pressure sensor, four soil moisture sensors, and an anenometer to calculate windspeed. Most anenometers, such as the Adafruit ADA1733, require a 12V input whereas the other sensors rely on a 3.3v or 5v input. As such, the circuit design (figure 12) was rather complex, requiring a buck converter and capacitor to step down the voltage for the smaller sensors.

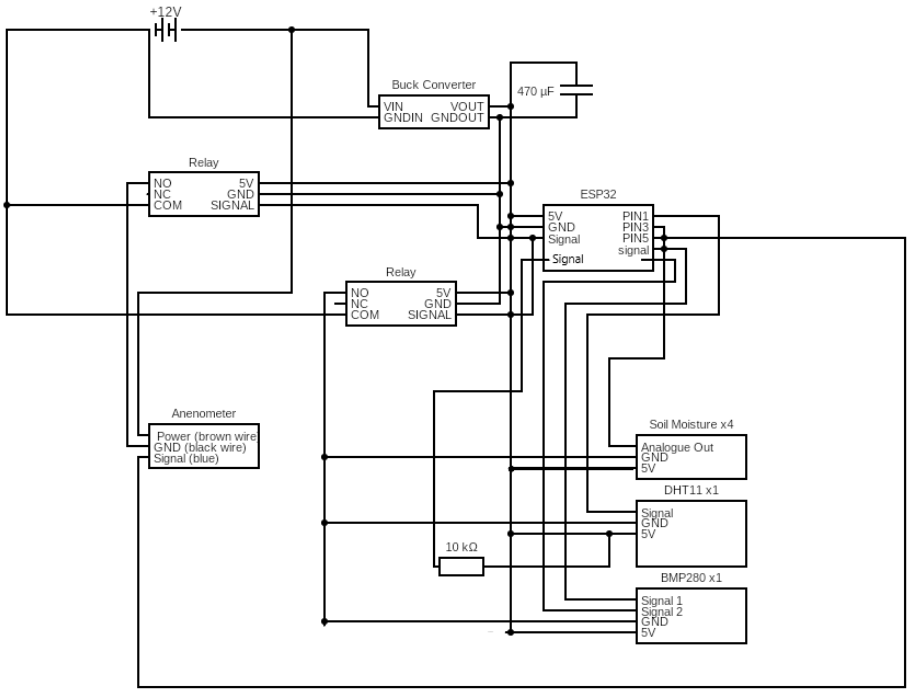

Figure 12: Original Circuit Design

Relays were used for the connections between the anenometer and soil moisture sensors. The relays were by default not allowing current to pass to the devices; the anenometer relay prevented the sensor from draining a large amount of battery throughout the day when its readings would not be necessary. In contrast, the soil moisture relays were to minimise electrolysis. Capacitive soil moisture sensors are known to suffer from substantial corrosion. One reason for this is

simply the exposure to the elements and to high PH soil, but another cause is the electrolysis that occurs when a voltage is passed through the sensors. By keeping the voltage separate from the sensors, the electrolysis is minimised and the lifespan of the sensors extended. To further extend the lifespan of the soil moisture sensors, the circuitry and connection at the top of the devices were taped with both masking and electrical tape. This minimised the water damage to the sensor while outside in the rain and being regularly watered.

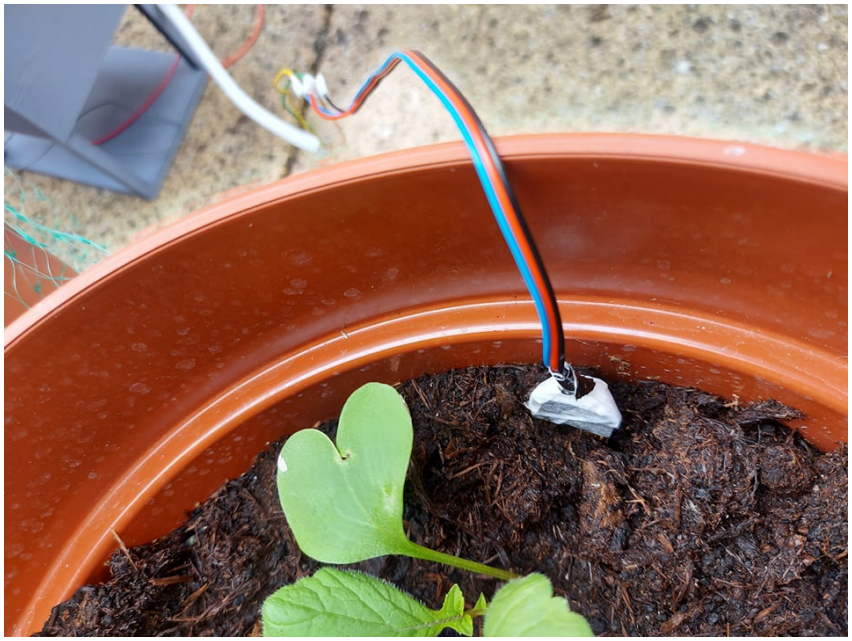

Figure 13: Soil moisture sensor taped and deployed

Once the networks were shown to perform without the need for wind data, the design was simplified with the 12v rail removed, as seen in the circuit shown in fig 14. This circuit was the final one that was built and deployed for the project, components were connected to the ESP32 either via soldering or jumper cables, depending on the sensor's connections.

4.3.2 3D-Printed Housing

As the weather station was to be deployed outside, a 3D-printed case was created to protect the circuitry from the elements. The design had several requirements: it must house the circuitry in a waterproof container, it must be elevated to prevent surface water damaging it, and finally it must allow connections to leave the housing so that it may deploy sensors outside of the housing. A close of up of the final design may be seen in figure 15, and the system (bar

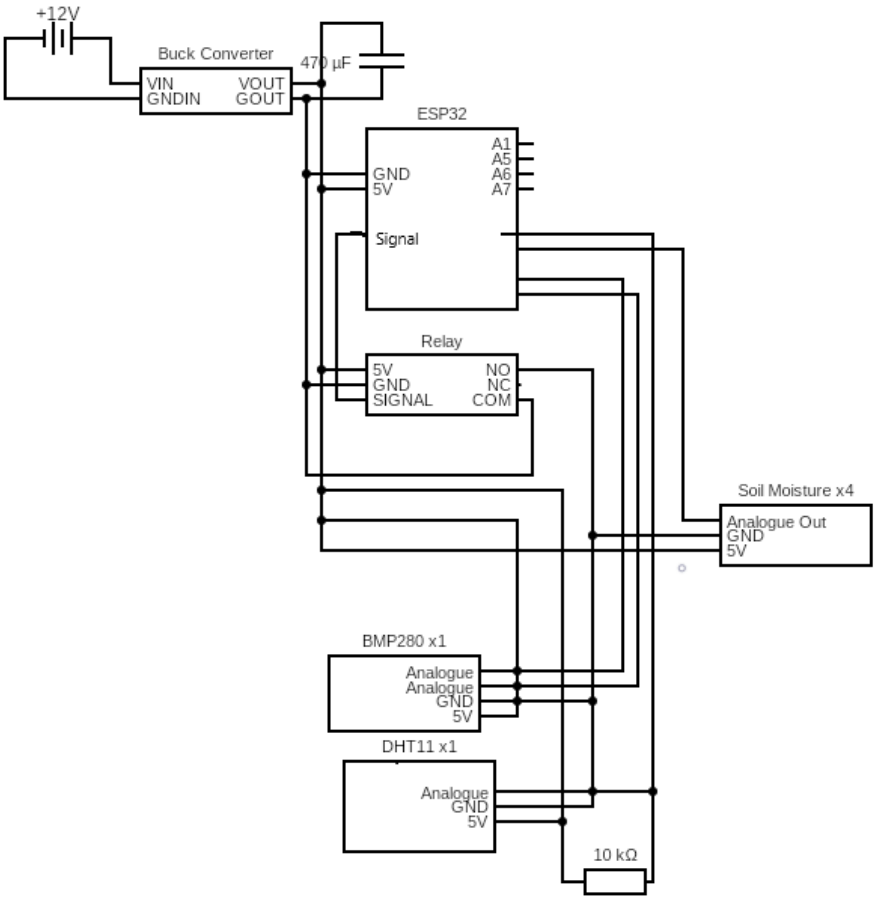

Figure 14: Final circuit design without anenometer or the associated relay

the battery) fully deployed may be seen in figures 16, 17. 3D printing was used as it allowed for cheap production of custom designs (created using Fusion 360) using waterproof material. The design was printed in 4 parts. Firstly the legs and base were printed together, these are raised to prevent the circuitry box from sitting in surface water. This also offers the option for weights to be added on top of the base in heavy winds to prevent displacement. Next, the walls of the box were printed together; inside the box there is a series of notches. One notch allows the lid (printed separately) to slide in and out, creating a watertight seal but also allowing the lid to be removed, in figure 16 the lid is be observed to be slightly open, to highlight the sliding implementation. The lid is sloped, to allow rainfall to run-off. The final part is a thin plastic bed that the components fit into, which is slotted into the box and rests on a lip.

The sensors leave the box from a circular hole in the circuit bed, allowing the sensors to sit outside the housing while maintaining the waterproof protection for the circuitry. While the soil moisture sensors are implanted into the soil, the DHT11 and BMP280 are suspended below

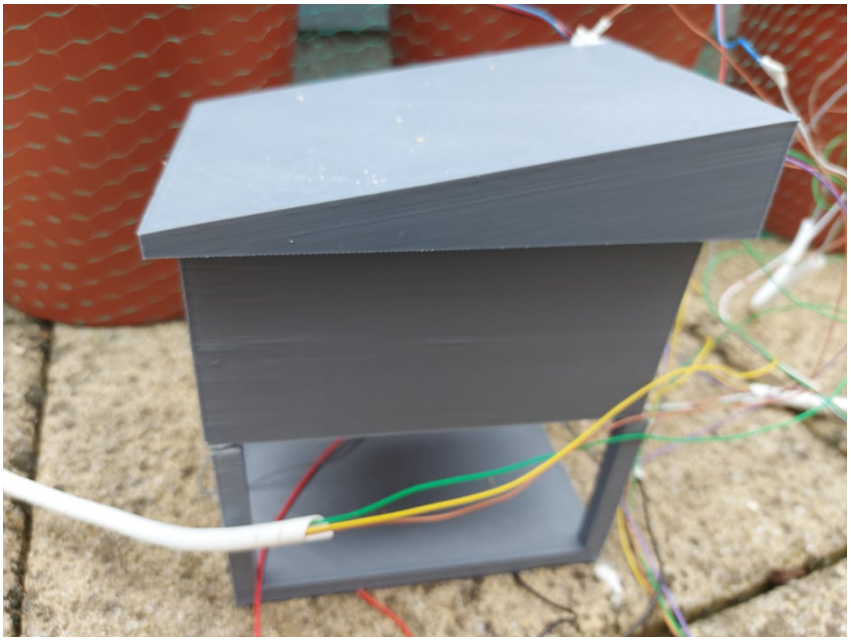

Figure 15: Close up view of the deployed weather station

Figure 16: Deployed weather station with the potted radishes

the station. This allows them to be protected from the weather, while still able to take readings outside which is imperative as the temperature, humidity and pressure will be different inside the box than outside. This is due to a range of factors, including heat from the circuitry altering the conditions. The suspended sensors are highlighted in figure 18

Figure 17: Deployed weather station with the potted radishes, overhead view

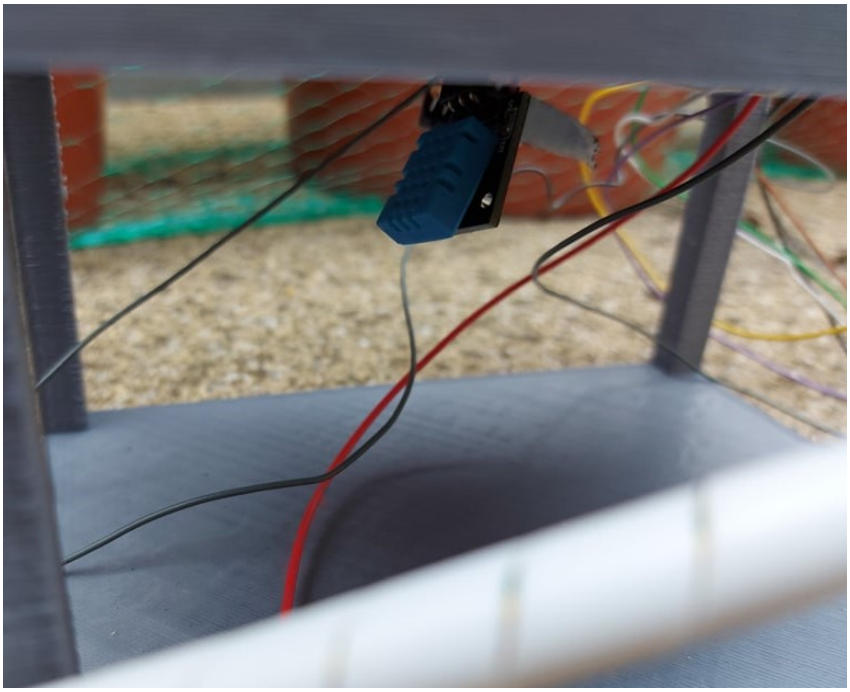

Figure 18: DHT11 and BMP280 sensors deployed outside of the box to ensure accurate readings

4.3.3 Software

The software for the weather station was written in the Arduino programming language. The ESP32 works as a server, accepting clients and passing their requests to the relevant endpoints. Each sensor has its own endpoint, as well as a "get all" end point for convenience. When a request is received, the relays are turned on and the sensors are queried; the values are calculated and sent back to the clientside, before the relays are switched off. This architecture was chosen as it is extensible and scalable, it is agnostic to the number of sensors it contains, as well as to what device is querying it. Furthermore, in a multi-agent system ESP32's could in theory be used to communicate with each other in this architecture, so the design is future proof for whatever extensions may be added in the future. The BMP280 was connected to the ESP32 via an I2C connection, the DHT11 uses serial data and the soil moisture sensors used analog reads (where the sensor sends a voltage to an analog pin which is then interpreted as a value).

The dew point is not something that can be directly measured, and so is calculated on the ESP32 from the temperature and relative humidity. The equation is given below:

$$Ts = (b\alpha(T, RH))/(a - \alpha(T, RH)) \tag{16}$$

$$\text{where: } \alpha(T, RH) = ln(RH/100) + aT/(b + T) \tag{17}$$

here, Ts is the dew point, T is the temperature, RH is the relative humidity and a and b are coefficients:

$$a = 17.62 \tag{18}$$

$$b = 243.12 \tag{19}$$

The client side is written in Python3. The client is implemented using an OOP framework, and has 3 primary functions. Firstly, the client connects to the ESP32 and requests the updated data (saving it to file). Next, the EWC network is trained according to the new data and the performance of both the EWC LSTM and the base LSTM is logged. Finally, both networks then make their predictions on the next 24 hours. This system is streamlined and largely autonomous, requiring very little of the human user.

4.3.4 Deployment Issues

There were several issues that occurred during deployment of the hardware. First of all, the buck converter works by outputting a ratio of the input. This precise functionality was not known during development and, as it had previously been used for a project with 5v input, it

was believed it would output 5v. However, as it outputs a ratio, and the ratio was set as 1:1, the buck converter allowed the full 12v through the circuit, destroying both the ESP32 and BMP280. Due to this, deployment time was delayed due to shipping. During this time, public weather forecasts from the UK Met Office and a handheld soil moisture sensor were used to collect the weather and soil data.

One issue with the deployed hardware was that sometimes the sensors would disconnect. This was due to the poor wiring that would lead to interference or jumpers coming apart. The poor wiring can be seen in figure 19. This was solvable in the short term by opening the housing and reseating some of the wires. However, a longer term solution is to print the circuitry to a PCB and use that instead of open wires. This would ensure the wiring is robust and that it wasn't going to move or disconnect. This was only a minor issue that occured rarely, but it is an area for improvement.

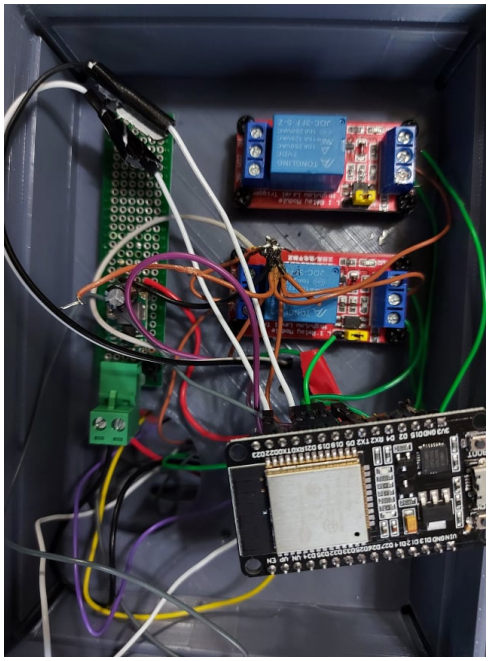

Figure 19: Inside of the Weather Station

Another issue that occurred post deployment was that the soil moisture sensors were in practice rather poor. This is likely due to their cost, but the sensors would consistently return a maximum reading for all pots, due to either degradation of the sensors or just low quality hardware. As such, it was sometimes necessary to use the handheld sensor which offered a greater degree of precision. This was undesirable as it undermines the concept of using sensors to automate the task, but as the problem was avoidable given sufficient funding, it was deemed more important to obtain accurate data. The sensor system was shown to work, it simply needs slightly higher

quality components.

4.4 Variational Bayes for Learned Weight Constraints

Once the system was deployed and the radishes were growing, an extension to elastic weight consolidation was explored. This was not used in the deployed system, but may be used for future work.

4.4.1 Theoretical Grounding

While EWC works well, estimating the likelihood term $p(\theta|a)$ using the Laplace approximation may be too simple to appropriately model certain distributions. As discussed in section 2.5, this approximation is limited and may be improved. The Laplace approximation is estimating the likelihood term by sampling from the following distribution (extracted from the Fisher information matrix):

$$p(\theta|\mathcal{A}) \sim \mathcal{N}\left(\theta^*_{\mathcal{A}}, \left(-\frac{\partial^2(log(p(\theta|\mathcal{A})))}{\partial^2\theta}\bigg|_{\theta^*_{\mathcal{A}}}\right)^{-1}\right) \tag{20}$$

The mean is the point of maxima of the distribution, and the variance is the inverse of the fisher information matrix, the importance of which is outlined in the related work section 2.5.

Variational autoencoders are predicated on variational Bayes', using binary cross entropy loss and the Kullback-Liebler divergence to find the best approximation of a distribution $p(x|z)$. By providing the network parameters as input, such that $x = \theta$, and by conditioning z on the data of a given task, a, it may be deduced that the variational autoencoder is approximating the probability density function $p(\theta|a)$.

VAE models have been used to calculate intractable likelihood terms in Bayesian problems before, such as in Choi et al. (2020). To the best of the author's knowledge, however, the application of this to estimate the importance of network weights for a task is a novel contribution to the field.

4.4.2 Implementation

In order to predicate the latent space, z, on the task, a mask is applied. As in Choi et al. (2020), the class is represented by the absence of parts (i.e. set parts of the latent space to 0 based on the presence of a class). The predicted class is then calculated as the lowest valued vector in the latent space:

$$\hat{y} = argmin[\sum_{j \in C_1} z_j^2, \sum_{j \in C_2} z_j^2, ..., \sum_{j \in C_{N_c}} z_j^2] \tag{21}$$

This constraint is enforced with the class loss function:

$$L_{class} = \text{cross\_entropy}(softmax(-w), \hat{y}) \tag{22}$$

Adding this to the VAE loss gives the final loss function:

$$loss = BCE(x, \hat{x}) + KLD(N(\mu_x, \sigma_x), N(0, 1)) + \lambda L_{class} \tag{23}$$

where lambda is an importance hyper-parameter. Once the network is trained, the μ and σ parameters in the network may be sampled for each input, meaning that the sample is an approximation of the likelihood $p(\theta|a)$.

One issue is that inputting a single θ does not work, as then the latent variable cannot be a lower-dimensional representation of the value, and reconstruction would just be a 1-to-1 mapping. Instead, each time the elastic weight consolidation is called, the set of weights in the classification network are randomly sampled without replacement, yielding x weights to be passed as input to the VAE; x is selected such that there exists a value y where $x \cdot y = n$, where n is the total number of parameters in the classification network. The sampling is then repeated y times so that every weight has been passed through the VAE. This means that the calculated likelihood is averaged across the sampled weight vector x. The sampling has benefits and disadvantages. The disadvantage is that for a given epoch each weight will not have an exact likelihood for itself, rather it will be averaged and so useless weights may be maintained for longer than they should be, and important weights may be changed more than they should be. However, given enough iterations, the important weights will on average be better protected, as the sampling is redone every epoch and thus the presence of the other weights can be considered noise sampled from the distribution of network weight importance. The advantage of sampling is that randomness is commonly a crucial component of learning. For example, stochastic gradient descent, genetic algorithms, novelty search and simulated annealing all rely on randomness to

explore search spaces and overcome local minima. The added noise may allow the network to move out of local minima and towards alternative solutions. To elaborate, treating lifelong learning as a multi-objective optimisation problem, there exists a Pareto frontier of optimal solutions that are objectively superior to all other solutions. As EWC heavily constrains the search space, it is entirely possible that Pareto-optimal solutions may be inaccessible. By adding the sampling, the noise may allow these constraints to be looser and for better solutions to be found. However, it is also possible that the sampling could simply lead to catastrophic forgetting and reduce network performance.

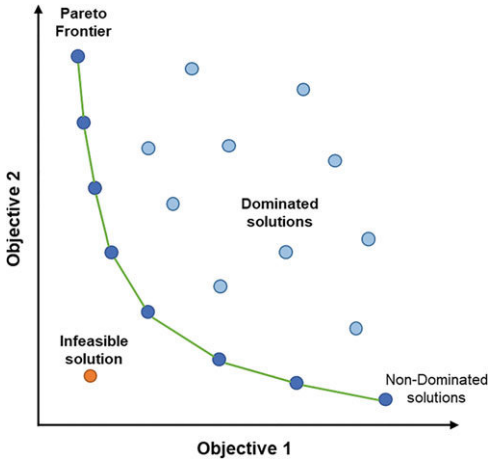

Figure 20: Pareto frontier for a multi-objective optimisation problem. Source: Liu et al. (2020)

The training of the VAE was conducted prior to each new task being presented, and then the VAE likelihood estimation was implemented to replace the Laplace approximation inside the EWC code. Otherwise, the EWC process remained the same. Once implemented, a grid search was conducted to calculate the most appropriate hyper-parameters.

5 Results

For all statistical tests conducted, a confidence threshold of 0.05 was required in order to reject the null hypothesis.

5.1 Regression

5.1.1 Experimental Setup

For the regression task, the five selected neural models were trained for 40 epochs on the MeteoNet dataset with a batch size of 43. Each network was trained 3 times, and the median result used for evaluation. Early stopping was used where necessary. The CNN and LSTM models used the Adam optimiser with a learning rate of 0.001, while the MLP used stochastic gradient descent with a momentum of 0.9.

5.1.2 Analysis

Figures 21 and 22 show the train and test set loss for all evaluated models for the precipitation regression task. Despite several models being evaluated, as well as a grid search for each being conducted, no architecture was able to obtain an RMSE loss of less than ~ 1.90. Interestingly, the CNN and MLP performed considerably better than all of the LSTM models, with the CNN being the best overall.

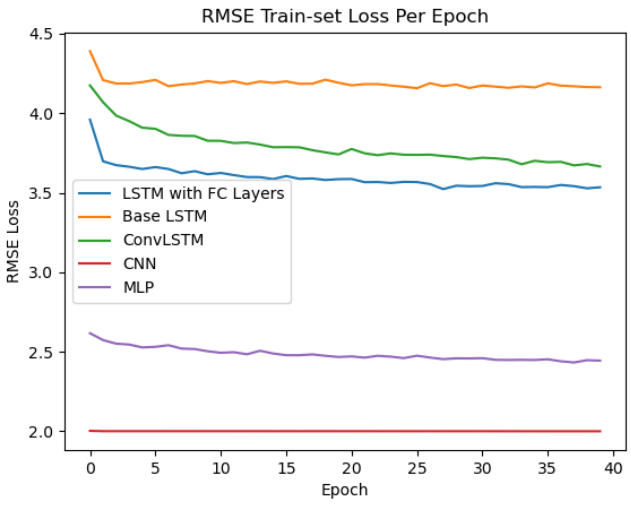

Figure 21: RMSE train-set loss for precipitation models

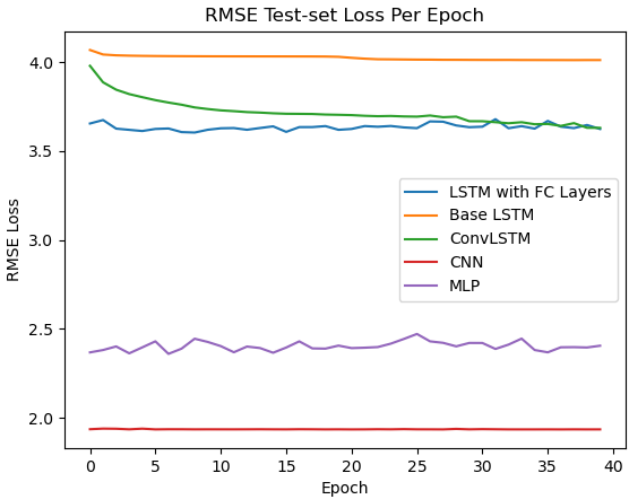

Figure 22: RMSE test-set loss for precipitation models

Oddly, the test set loss was slightly better for the models than the train set loss. This, combined with the overall poor performance, indicates that the problem was not that of overfitting, but rather the models were unable to adequately learn the task. One cause for this could be the substantial data imbalance leading to the vast majority of samples residing to values under $\sim 2.5mm$ rainfall. As such, failure to estimate days with large rainfall (up to $60mm$) could lead to large errors due to the networks only learning to model low-rainfall data. Another possibility is that the models lacked the complexity to adequately encapsulate the problem. If the networks were too shallow or too simple, then there is a hard boundary at which point they cannot learn anymore information and thus their performance would be gated. The final possibility is that the problem is very difficult, and that the input vector does not possess the necessary information to accurately estimate the amount of precipitation for the next 24 hours.

Figure 23 shows the distribution of the daily precipitation against each input variable. At an eye test, there appears to be correlation in at least two of the variables. Spearman's correlation coefficient was calculated for each of these distributions, the results are shown in 1. Spearman's was chosen over Pearson's as it does not assume a normal distribution of both datasets. Values close to 0 indicate no correlation, whereas values approaching ± 1 indicate a strong correlation. The test shows that the humidity and pressure have a statistically significant positive correlation of $\sim 19\%$ and $\sim 40\%$, respectively, with dew point boasting a very slight correlation and temperature offering no correlation. These correlations are not so large that linear regression could predict precipitation, validating the choice of using neural networks. The correlation is, however, large enough that the models should be able to solve the task to a reasonable stan-

dard. As such, the models were changed to instead classify the rainfall, as this may require less precision and thus may be easier to accomplish.

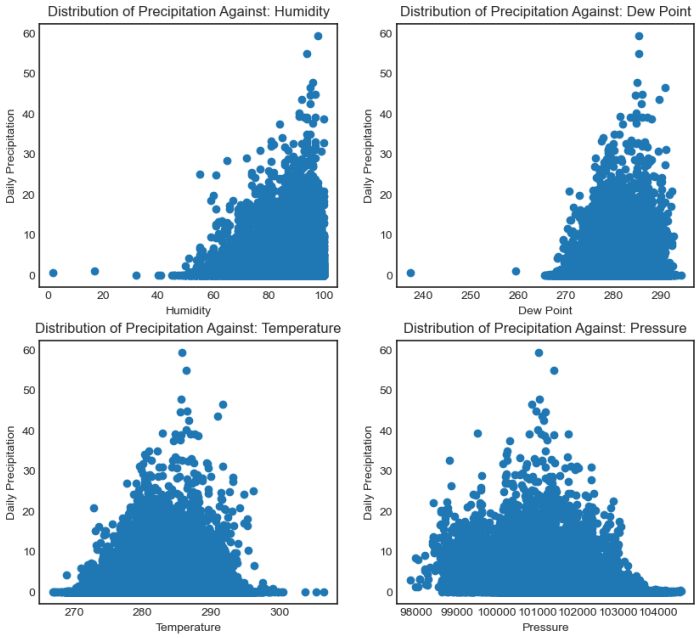

Figure 23: Distribution of daily precipitation against each input variable

Spearman's Correlation Against Precipitation		
Variable	Spearman's Correlation	P-value
Humidity	0.188	$1.140e^{-140}$
Temperature	-0.003	0.668
Dew Point	0.051	$9.279e^{-12}$
Pressure	**0.405**	**0.000**

Table 1: Spearman's correlation coefficient for each variable against daily precipitation

5.2 Classification

5.2.1 Experimental Setup

Each model was trained five times, with the comparison then being drawn between the median run for each model. Training was undertaken for 100 epochs with early stopping, with a batch size of 43. The time-series LSTM model was trained with the Adam optimiser and a learning

rate of 0.00001. The remaining LSTM models used a learning rate of 0.0003. The CNN also used Adam, but with a learning rate of 0.0001. The MLP used stochastic gradient descent, with a learning rate of 0.005 and momentum of 0.9. As a note, a time-series CNN was also tested, but performed substantially worse than the original CNN and so was not explored in detail, in favour of further optimising the base CNN model.

5.2.2 Analysis

When evaluating the performance, using a simple accuracy metric was insufficient, due to the massive class imbalance. If a model guessed only the majority class for every sample, it would achieve an overall accuracy of $\sim 72\%$. As such the mean F-score was used, which combines both the precision and recall values, as given in equation 24. The F-score was taken across both classes and the mean calculated, meaning that despite an unequal number of samples belonging to each class, each class had equal weighting in the final score. However, even this won't completely account for a model which can predict a very high accuracy in one class but not the other; but each class will at least hold equal weight.

$$\text{F-Score} = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{24}$$

Figure 24 shows the change in mean f-score over the training time. The LSTM and time-series LSTM both featured a single feedforward layer after the LSTM, to convert the output into a classification. While models were trained for 100 epochs, much of the improvement was made during the first 5-10, which is unsurprising due to both the large dataset size and the small number of input parameters. This meant a single epoch contained a huge amount of samples, and that there were fewer weights to train compared to something such as an image classification model. As such, early stopping was used when evaluating the best performance for each model. Figure 25 also shows the mean f-score, but without the first 5 epochs to better show model performance in the remaining epochs. As was the case with regression, the LSTM and ConvLSTM models both perform poorly. This is likely because without the time-series data, much of their strengths are redundant as they cannot utilise their long-term memory effectively. In contrast, once the LSTM was provided with time-series data, it became the best model evaluated out of all of the models. The MLP and CNN continued to perform reasonably, with the CNN offering a much lower variance per epoch and a higher overall performance. The time-series LSTM, however, remained the best performing model.

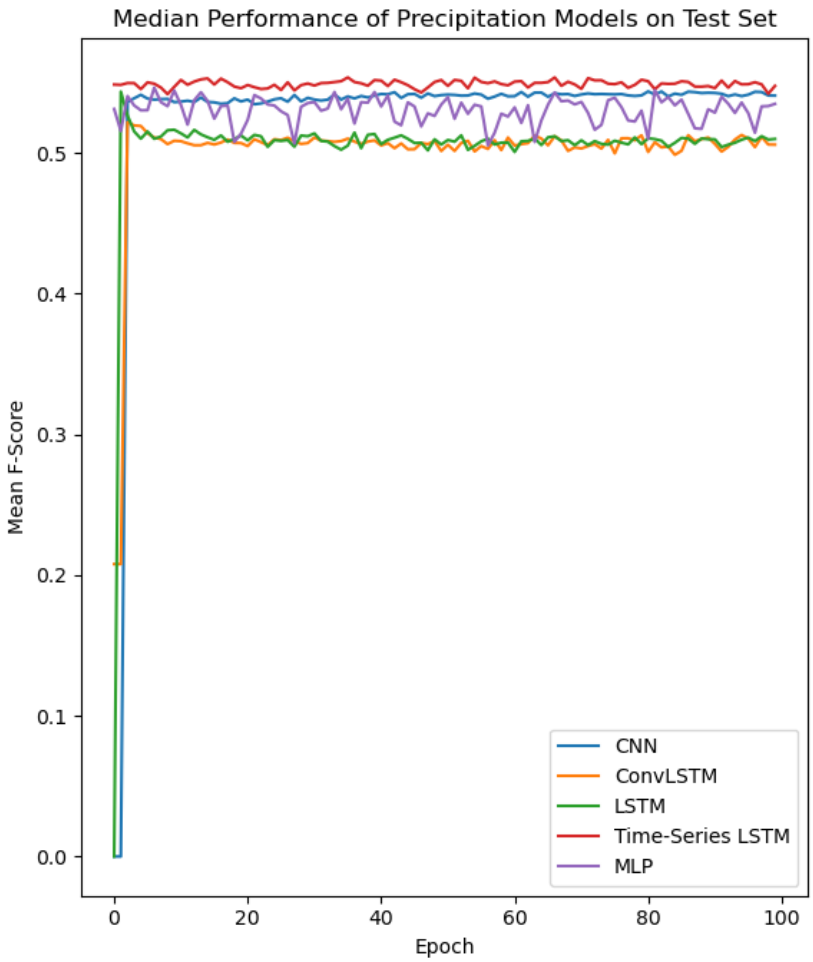

Figure 24: Median precipitation model performance on classification task

The data presented in table 2 explores the results in greater detail. The first observation made is that, despite the MLP and CNN appearing to perform well based on the F-score, both of the models still had a large bias towards the majority class, 'no rain'. Both achieved an accuracy of over 90% for this class, but then failed to reach even 45% for the minority class. As such, their results are in practice considerably worse than the time-series LSTM, which was the only model that maintained a near-even accuracy between the two classes, just a 5% difference. The base LSTM and the time-series LSTM had the lowest standard deviation of F-score across the 5 runs, indicating that these models were the most consistent in their performance. The time-series LSTM and the ConvLSTM both had the best mean recall, indicating that they were the best at correctly identifying correct samples. In contrast, the MLP model had the highest proportion of positive predictions actually correspond to a positive sample, thus yielding the highest precision. Despite this, the time-series LSTM was the best performing model overall, and also had the best balance between the two classes and was therefore selected as the model to take forward for the rest of the project.

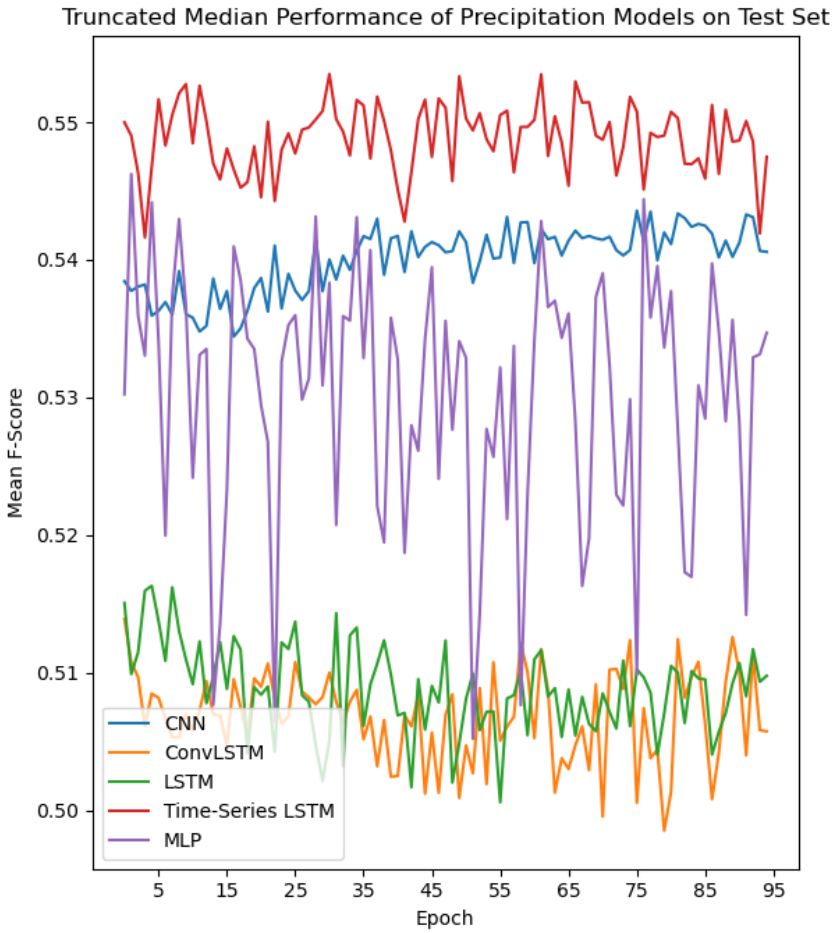

Figure 25: Truncated section of median precipitation model performance on classification task

	Model				
Metric	LSTM	Conv LSTM	MLP	CNN	Time-series LSTM
No Rain Class Accuracy	0.86	0.779	**0.926**	0.912	0.733
Rain Class Accuracy	0.53	0.636	0.4	0.422	**0.683**
Mean Precision	0.503	0.473	**0.516**	0.511	0.495
Mean Recall	0.70	**0.708**	0.663	0.667	**0.708**
Mean F-score	0.543	0.522	0.546	0.544	**0.554**
Standard Deviation of F-score	**0.002**	0.278	0.018	0.295	0.003

Table 2: Classification model performance metrics

To confirm the performances of each model were significantly different, an Alexander Govern statistical test was conducted on the mean F-scores. This test was chosen as unlike the one-way ANOVA test, this test does not assume homoscedasticity, instead relaxing the assumption

of equal variances. This is important as homogeneity of variances could not be assumed given that the standard deviation of model performance was different between each network. The test found that the models were different with $p = 1.06e^{-6}$. Comparing the base LSTM to the time-series LSTM found that introducing the historic data did improve the model significantly, with $p = 5.168e^{-8}$ and as such the null hypothesis is rejected in favour of the alternative, that using historic data did improve model performance. Finally, the performance between the CNN time-series LSTM was compared, to evaluate whether it was definitively the best model. Surprisingly, the test returned a P value of 0.166 and so the null hypothesis cannot be rejected. While the model's F-score may not be statistically better, the better class-accuracy distribution alongside the expected added value of historic data for aiding EWC learning with sensor-feedback, meant that the time-series LSTM remained the favourable model. The best time-series LSTM model, not the median model, was selected for deployment. This model achieved 72.3% accuracy on the 'no rain' class, and 72.4% accuracy on the 'rain' class. As this problem is non-trivial, a total accuracy of $\sim 72\%$ can be considered a successful and effective prediction network.

5.3 Lifelong Learning

5.3.1 Experimental Setup

When evaluating the efficacy of EWC for transferring precipitation prediction, both the standard time-series LSTM model and the EWC version were trained 9 times, with the median run being taken and evaluated. The models were presented with three tasks: MeteoNet precipitation prediction, MeteoNet wind direction prediction, and NCAR precipitation prediction. The models trained for 30 epochs on the first task, before the second task was introduced and training stopped for the first. After 30 epochs training on the second task, the third task was introduced and training on the second ceased. After each epoch, the test loss was evaluated for all tasks currently viewed. This means that the test performance on the first task was recorded for 90 epochs, the performance for the second task for 60 epochs, and 30 epochs for the final task. A batch size of 128 was used, and a sample rate of 200 for the EWC algorithm. Both models used the Adam optimiser with a learning rate of 0.00001.

5.3.2 Analysis

The results for each task can be seen in figures 26, 27 and 28, the error bars represent one standard deviation. The mean F-scores of each task may be seen in figures 29, 30, 31. The efficacy of EWC is highlighted best when viewing graphs for task 0 (the first challenge). Every time a new task is introduced, the standard LSTM suffers a significant loss in performance on the original task. In contrast, the EWC network suffers almost no loss in performance after the second task is introduced, and then while performance does decrease after the third task is introduced, it is better than the base LSTM. As the performance of the first task gets significantly worse when the third task is introduced, there is strong evidence to support the assumption that transferring from one region to another is problematic and can lead to a reduction in model performance, thus further validating the decision to leverage lifelong learning.

As the learning is constrained for subsequent tasks, it would be rational to assume that the EWC model would suffer performance degradation for the second and third task. However, by the end of the 90 epochs, the EWC network is equal with the standard network for task 2, and actually outperforms for task 3. What is interesting is that the model continues to improve for task 2; so despite being worse than the standard LSTM after the initial 30 epochs of training on task 2, exposure to the final task actually improves the EWC network's performance. Furthering this point, the performance of the LSTM does not degrade for task 2 when the final

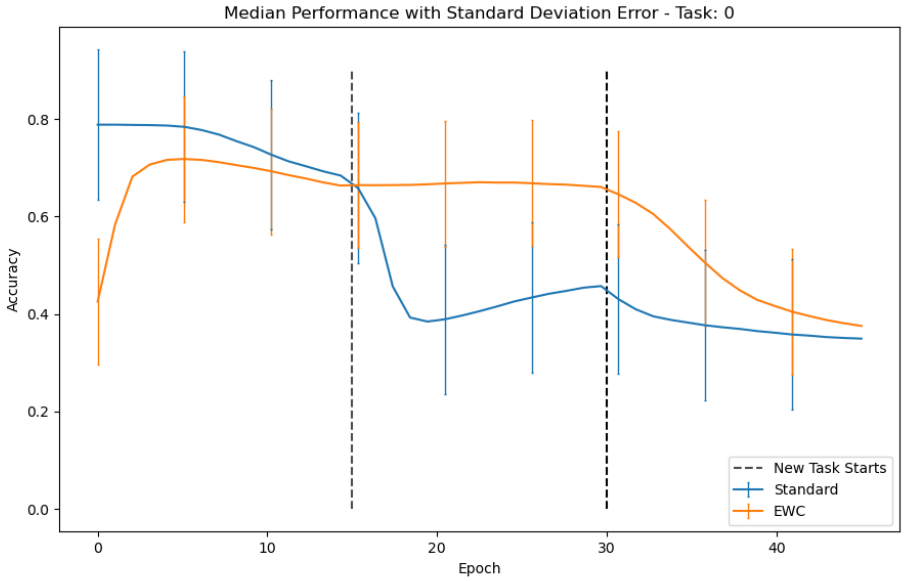

Figure 26: Median Run Accuracy for the MeteoNet precipitation prediction task

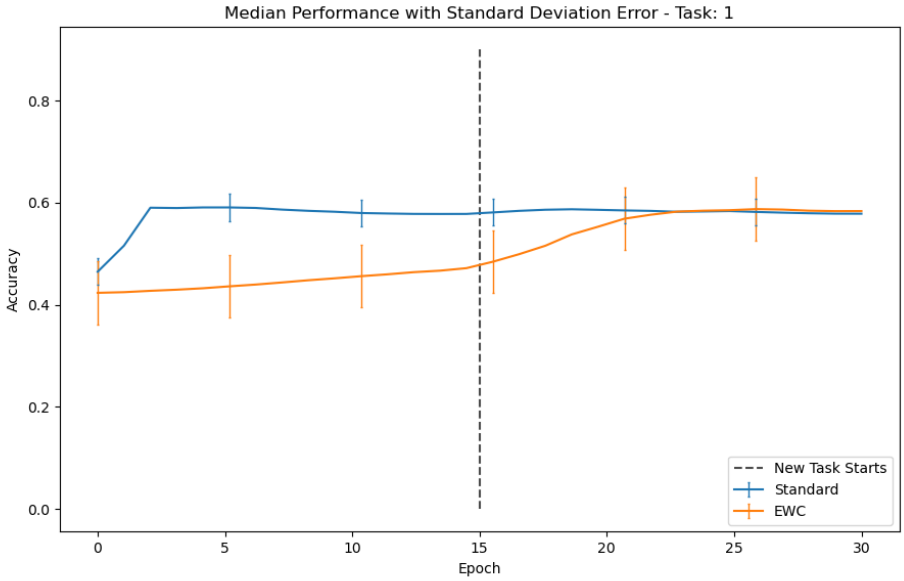

Figure 27: Median run accuracy for the MeteoNet wind direction prediction task

task is introduced. Combined, this suggests that the second and third task may share similar optimal weight configurations. The importance parameter has a large impact on EWC performance, and was set (along with other hyper-parameters) using a grid search. The problem with the importance parameter is that as it is EWC specific, literature is limited on what order of magnitude the parameter should take. A wide range of values were evaluated, and eventually the near-optimal value was found to be 1000000.

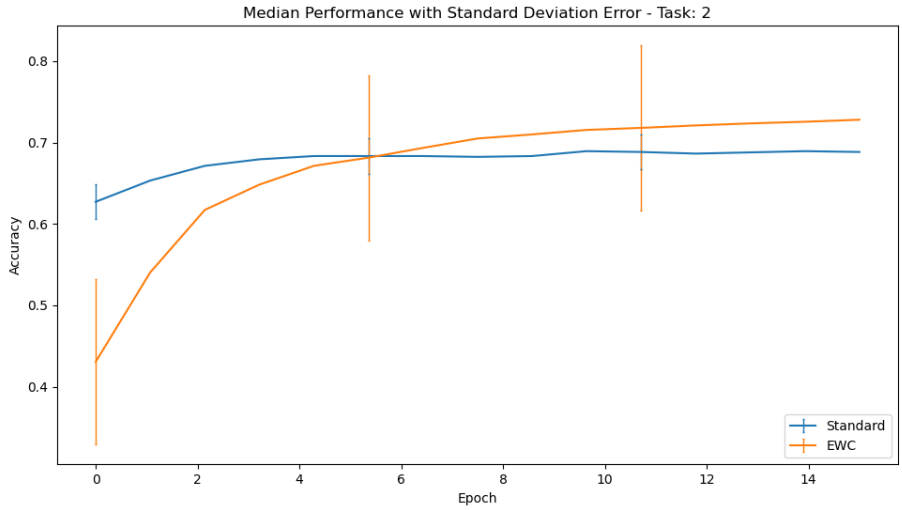

Figure 28: Median run accuracy for the NCAR precipitation prediction task

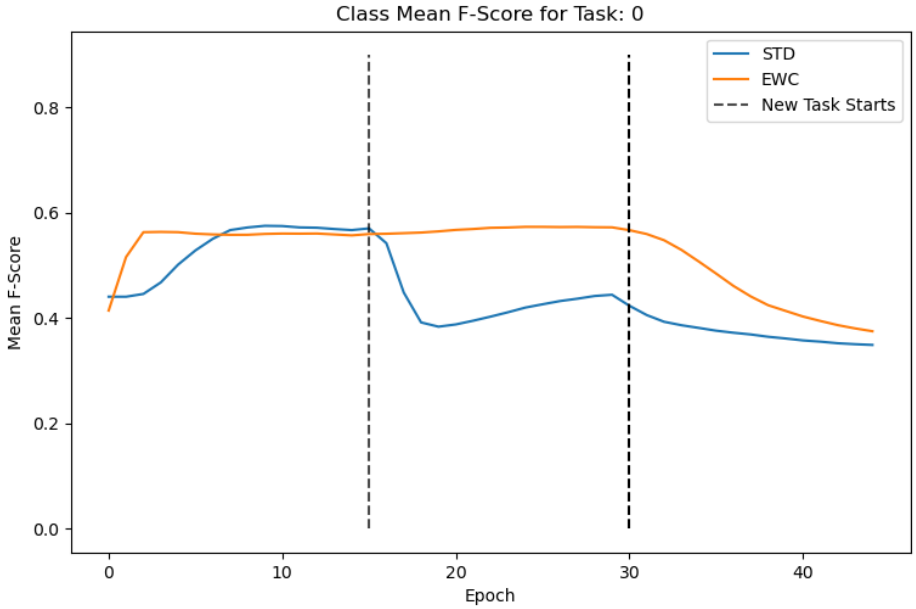

Figure 29: Mean F-score for the MeteoNet precipitation prediction task

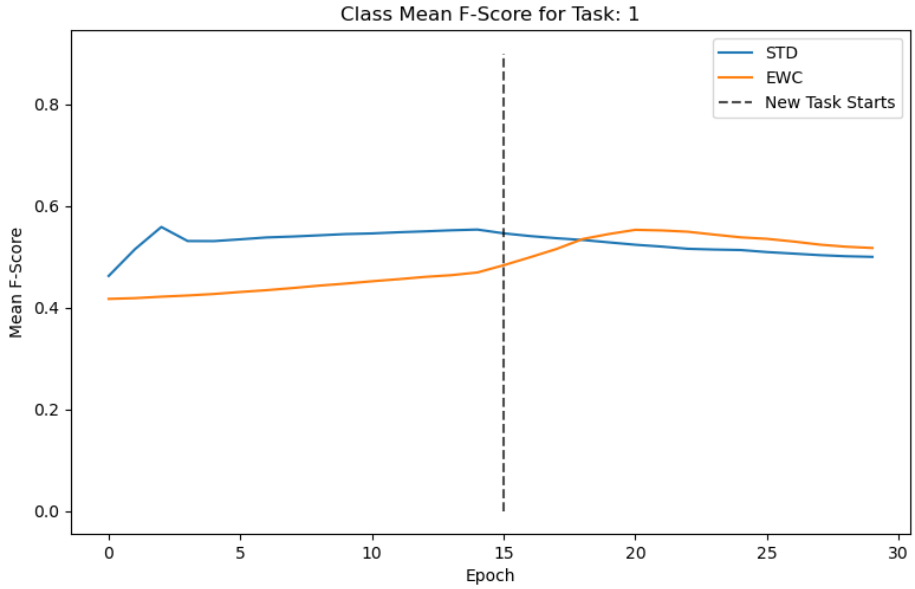

Figure 30: Mean F-score for the MeteoNet wind direction prediction task

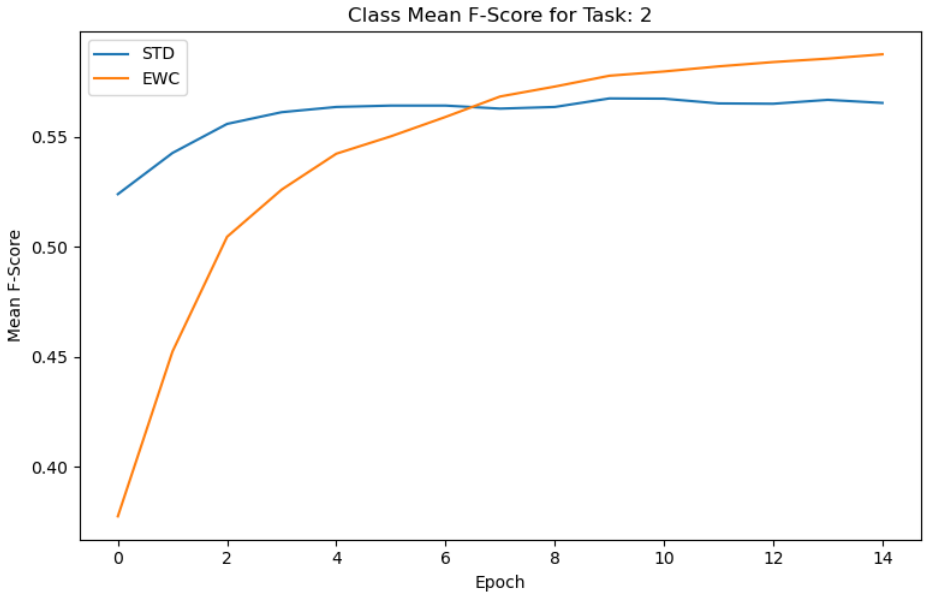

Figure 31: Mean F-score for the NCAR precipitation prediction task

To confirm that the EWC performance increase was statistically significant, a T-test was conducted for each of the tasks. The first task was found to have a significant difference in performance, with $p = 0.02$. This confirms the belief that utilising lifelong learning to facilitate the transfer of precipitation prediction capability is an effective strategy. For the other two tasks, however, the p-value was 0.49 and 0.56, respectively. This is interesting, as it indicates that despite the perturbations in performance between the two LSTM networks, their performance

cannot be said to be statistically different. As such, the EWC model is successfully improving classification accuracy for task 0 without sacrificing model performance for subsequent tasks, which means that for this task, EWC appears to offer the expected upside without compromising other aspects of the problem.

5.4 Real-world Deployment

5.4.1 Experimental Setup

The radishes were planted for 21 days, which is the lower-bound of the recommended growth time. This was due to project time constraints. To minimise the influence of extraneous variables during this time, the pots, the soil type, soil amount and watering amount were consistent across all test beds. Furthermore, all pots were placed as close to each other as possible, to minimise perturbations in exposure to the sun and rain.

5.4.2 Analysis

Table 3 shows the final radish yield for each of the methods. Two metrics were used to evaluate radish growth: mean leaf length and total field weight (i.e. the weight of all produce within the pot).

Radish Growth Metrics		
Method	Mean Leaf Length (cm)	Total Field Weight (g)
Human	8.25	15
Sensor Only	8.625	16
LSTM	**9.625**	**24**
EWC + LSTM	7.75	20

Table 3: Radish size and health metrics

The pot watered by a human agent was actually the worst performing of all pots. This pot had the lowest total field weight (15g) and the second lowest leaf length (8.25cm). This is interesting as it implies that average human agents are in practice not effective caretakers of crop and so urban agriculture could be significantly improved with the application of technology.

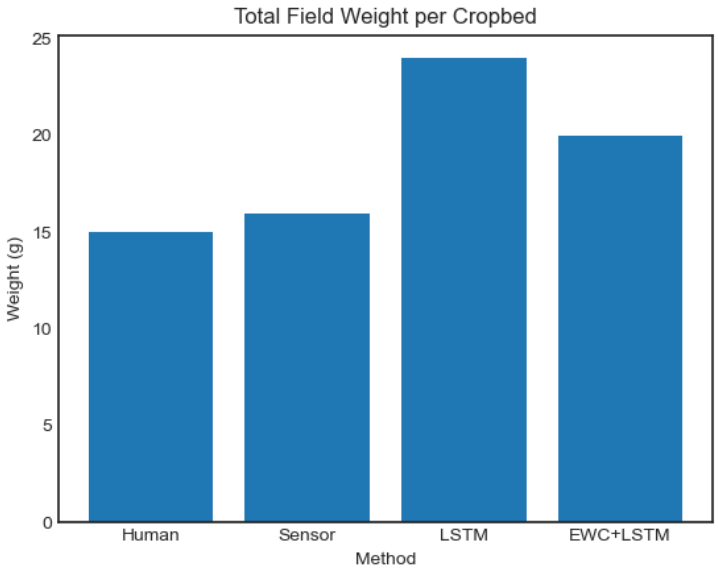

Figure 32: Total field weight of each cropbed

The sensor only pot had a total field weight of 16g, just slightly heavier than the human-managed pot. These values are so close that they could easily be attributed to variance or noise. The mean leaf length is 4mm longer, which is more significant than the field weight, but still fairly minor. As such, it appears that by using only a soil moisture sensor the human in the loop can be either matched or even improved upon when considering when to water.

The pot maintained by the base LSTM was by far the best pot. Achieving a total field weight of 24g and mean leaf length of 9.625, this strategy was a 1cm and 4g better than any other model. If the other LSTM pot is omitted, this method was 8g better than the competitors (human and sensor), representing a 50% increase in yield from the sensor pot. This heavily suggests that considering future states when deciding to water is of significant importance, as well as suggesting that the model built here was effective at doing so.

The EWC LSTM pot was somewhat of an anomaly. With all other methods, the total field weight and mean leaf length aligned, such that the crops with longer leaves had a greater yield. In contrast, the EWC LSTM under-performs and results in the worst mean leaf length (7.75cm), a full 0.5cm shorter than any of pot. However, the model also produced a crop yield of 20g, which was better than the human and sensor crop yields by 4g and 5g, respectively.

Figure 33 shows the decision making of both LSTM networks across the observation period,

while figure 34 shows the correctness of these predictions. There is a greater number of accuracy days than prediction recordings. This is because if the moisture level was beyond either the minimum or maximum threshold, the previous day would be recorded as an incorrect prediction and then the network would not be called as the situation was so extreme the prediction was redundant. For example, if the soil was exceptionally dry, it will need watering irrelevant of rainfall (bar very extreme rainfall events) and so the prediction was unnecessary. If the model did not make a prediction that day, then it would not be penalised or rewarded for the resultant soil moisture on the following sensor reading. Fortunately, this edge case handling was only necessary on 3 occasions. Combined with the cumulative accuracy graph in figure 35, these graphs illuminate that the performance of the EWC network was very poor early in the observation period, having an accuracy of just 33% after the first 3 days. Interestingly, it took only 1 day for the predictions of the base LSTM and the EWC LSTM to deviate. This indicates that maybe the learning rate was too large; which is not unlikely as it was increased by a factor of 10 for deployment as the original learning rate was fine tuned to accommodate a dataset for years of data. This notion is reinforced by the fact that between day 8 and 14 the EWC model cycles between correct and incorrect predictions, which means that the network parameters may be oscillating around a point of optima, as the learning rate is too high it cannot actually arrive at the optima. This is not unlike a PID controller oscillating around a desired point when the P gain is too large. As such, the EWC network performs worse than the base LSTM for the first 14 days of the observation period. However, during the last 7 days, the EWC appears to settle on a strong solution, as there are no incorrect guesses from the EWC despite two incorrect guesses from the base LSTM during the same period. This not only suggests that the EWC may have ended as the superior model, once the network adapted to the implicit encodings within the sensor-feedback, but it also may explain the discrepancy between the leaf length and yield weight. All other models were static (i.e. didn't learn) and so the development of the radishes will have remained fairly consistent. In the case of the EWC network, the early stages of development had worse conditions than later stages; as the leaves develop earlier than the edible swollen root, their development may have been hindered, but then the EWC model was better trained and performing well by the time the edible part of the crop began developing. Finally, a notable occurrence was that in the EWC pot a non-radish plant had also grown. Likely deposited by local wildlife (such as collection of birds whom are regular visitors), the additional plant will have added further competition to the four radishes, which may well have inhibited their growth. It is possible the EWC crops would have performed the best without this added strain.

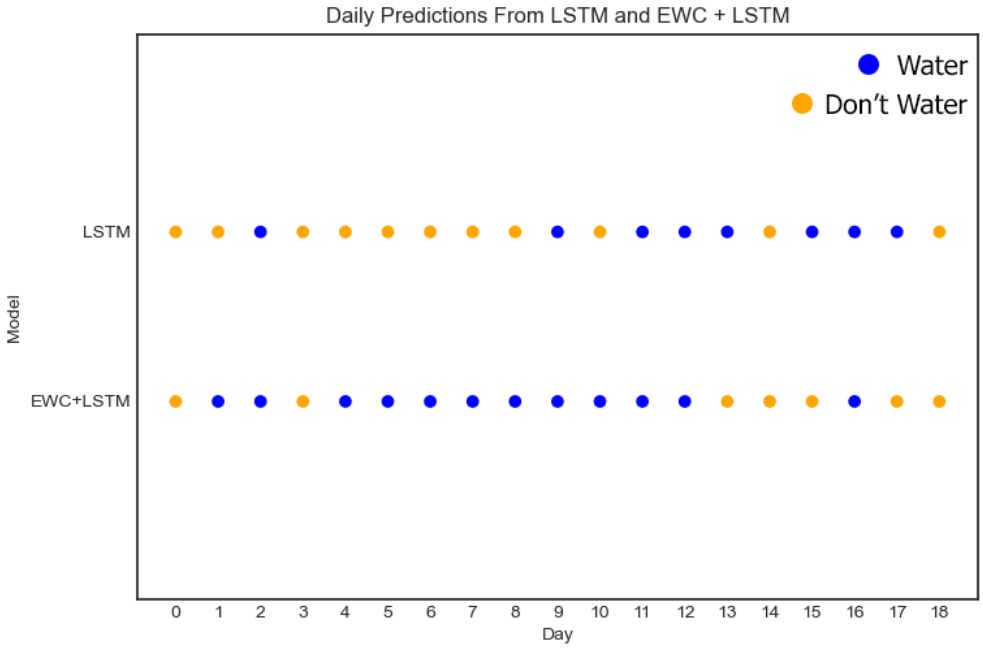

Figure 33: Soil-moisture predictions from both LSTM networks

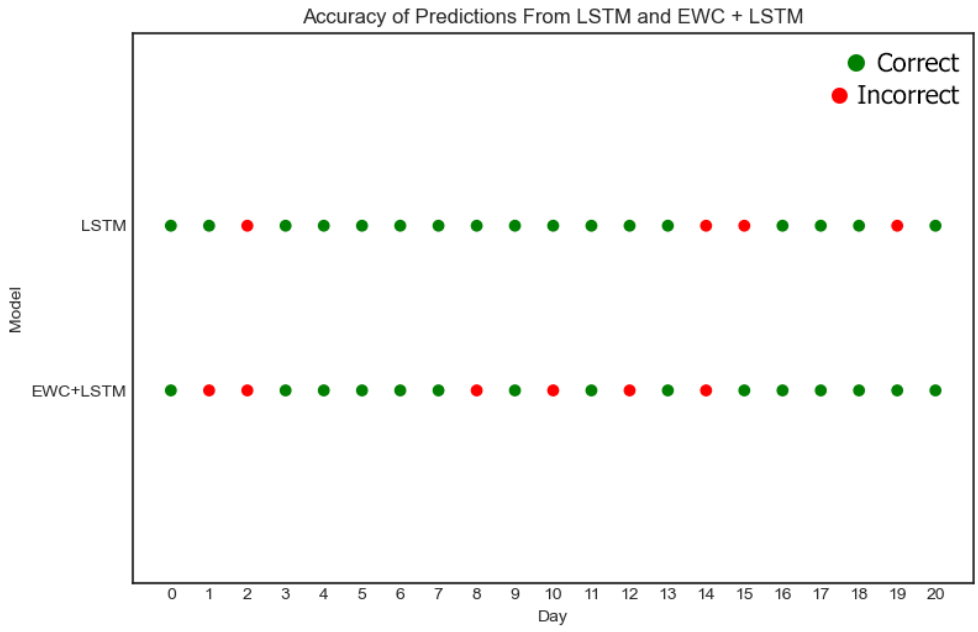

Figure 34: Accuracy per day of LSTM and LSTM with EWC models

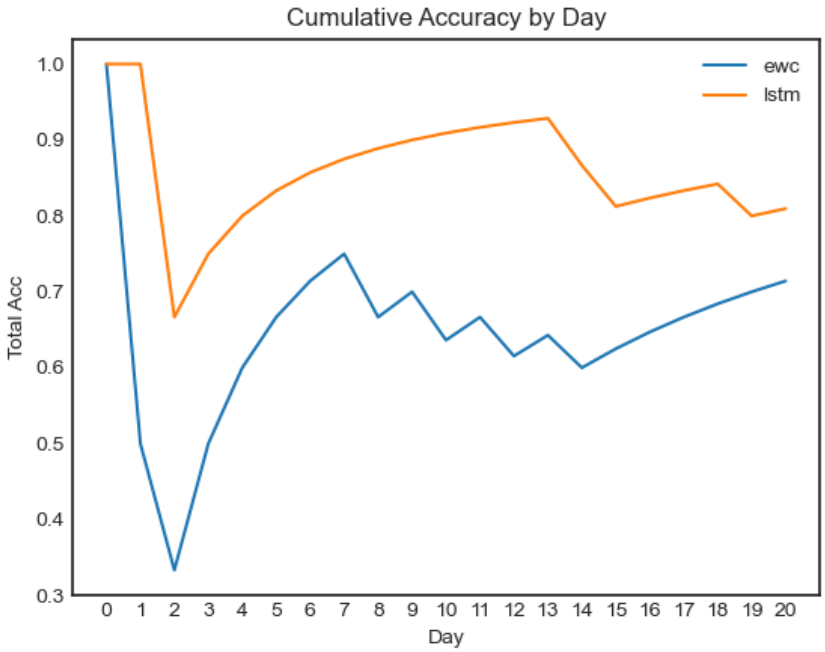

Figure 35: Cumulative accuracy over time of LSTM and LSTM with EWC models

While the crop growth metrics do suggest that the LSTM models are better than the other methods implemented, crop values alone are subject to large amounts of variation (especially over a single harvest). As such, the soil-moisture values were recorded for evaluation too. In the event of issues with the radishes themselves, the soil-moisture values will have been unaffected. Table 4 shows a set of descriptive statistics for the soil-moisture values, evaluated at their ability to minimise variance while maximising the proximity to a value of 575, the median of the target range $[550 - 600]$. Minimising the variance is desirable as it offers consistency for the growth of the plant, and maintaining the optimal soil-moisture will allow for better growth. The statistics show that the EWC network had the best mean and median soil-moisture value. This further supports the notion that this was overall the best method deployed. A weakness of this approach was that it had the largest variance of all the methods, although this is not unexpected when the model is actively changing during the observation period. In contrast, the EWC network's kurtosis was the least platykurtic; that is to say, the EWC's distribution had the largest peak.

The LSTM model also performed very well according to the descriptive statistics, outperforming the non-LSTM models with respect to average soil moisture, but once again possessed a larger standard deviation.

Soil Moisture Over Total Observation Period					
Method	Mean	Median	Std. Dev.	Skew	Kurtosis
Human	500	520	146	-0.244	-0.202
Sensor Only	501.524	540	**134.955**	-0.579	-0.532
LSTM	554.048	600	155.609	-0.653	-0.313
EWC + LSTM	**564.143**	**598**	170.5	-0.724	-0.106

Table 4: Soil Moisture Metrics

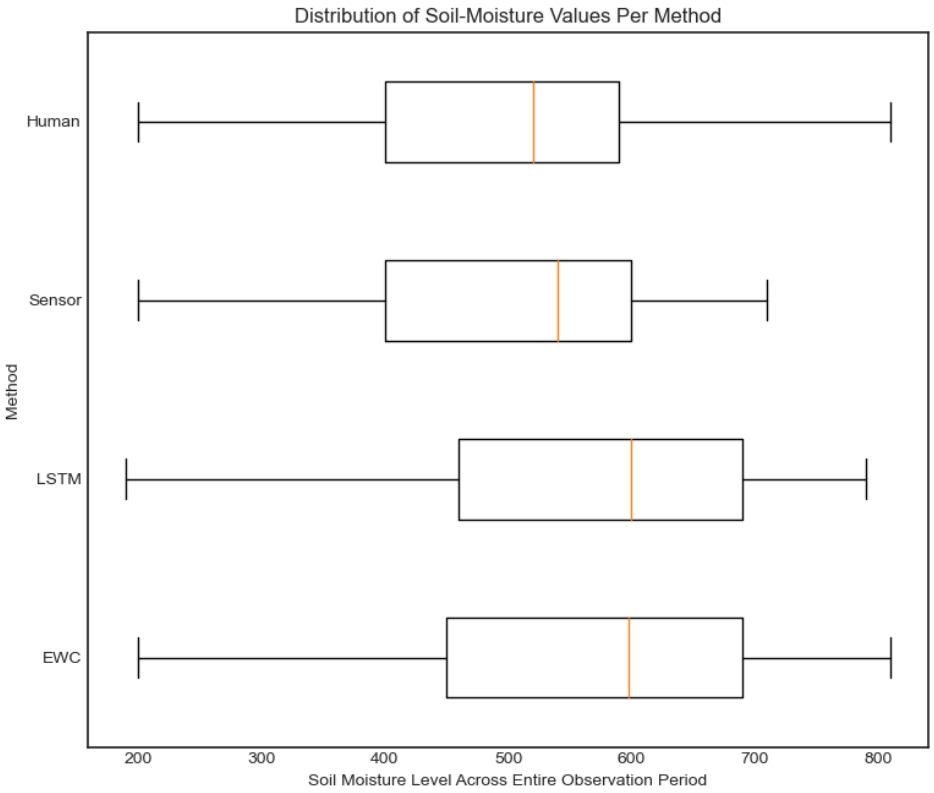

Figure 36: Soil-moisture distribution over entire observation period

Almost all empirical evidence implies that the LSTM models are considerably better than their counterparts, supporting the notion that predicting future states is a vital step in optimising soil-moisture. However, empirical evidence is insufficient and thus an Alexander Govern statistical test was conducted. Unfortunately, the test returned a p-value of 0.399 and thus the null hypothesis cannot be rejected. While this means the LSTM models cannot be defined as objectively better, the statistical test may be influenced by the many factors that influence soil-moisture outside of simply the watering regime. This is highlighted in figure 37, where

all models follow a similar trend as the moisture levels are heavily influenced by the weather conditions. However, the models still offer value as they allow the soil-moisture to be optimised as much as possible under the constraints imposed by the elements. Over a longer time period, it is likely that the influence of the model would become more evident and statistically robust. Furthermore, repeat runs with more crop would allow for the statistical analysis of the crop yield, which from this experiment seems to be heavily improved (up to 50% improvement) by the choice of optimisation model selected. Both LSTM models outperformed the non-LSTM models, with the base LSTM yielding better produce and had fewer incorrect watering decisions, while the EWC network maintained a better average water moisture and evidence suggests it may have stabilised and thus become better than the base LSTM in the later stages of the experiment.

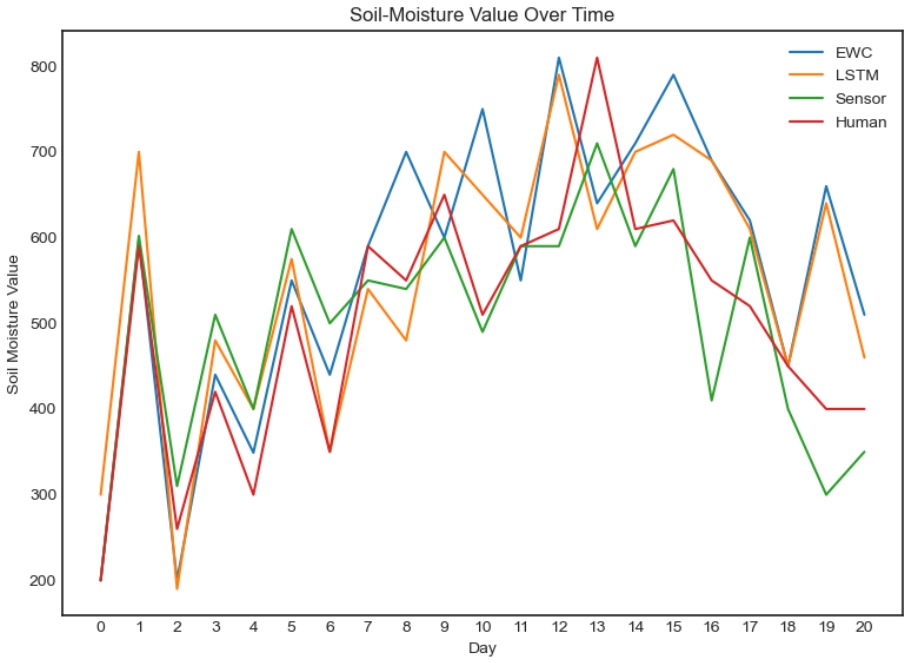

Figure 37: Change in soil-moisture over entire observation period

5.5 Variational Bayes for Learned Weight Constraints

5.5.1 Experimental Setup

The final contribution of this work was an introductory study into the efficacy of using a VAE to estimate the intractable likelihood term $p(\theta|a)$ within the EWC algorithm. This was accomplished by replicating the permutation MNIST task with an MLP as conducted in the original EWC paper (Kirkpatrick et al., 2017). Each model was run 9 times, with the median being obtained and used for evaluation. The performance of a network to obtain the median was defined as the mean final accuracy across all tasks. Error bars on the graphs represent a single standard deviation. The sample rate for the lifelong learning models was maintained at 200 samples, to control extraneous variables. For the remainder of this section the VAE model refers to the MLP network informed by the VAE sampling paradigm, not the VAE itself. Training the VAE on top of the MLP is somewhat time consuming, and so there was not time for an extensive grid search. After a small search, a VAE learning rate of 0.0000035, a VAE lambda value of 0.0005, and an importance of 10000 were chosen. The EWC used an importance of 1000 and the MLP had a topology of 2 hidden layers of width 400, ReLu activation functions, stochastic gradient descent and a learning rate of $1e^{-3}$.

5.5.2 Analysis

The performance of each model is shown in figures 38,39,40. For the original MNIST data, task 0, the EWC outperforms both the standard and VAE methods once the first permutation task is introduced. However, once the second set of permuted images are introduced the performance of both EWC and the standard model deteriorates. In contrast, the VAE model performs comparatively to the standard network for task 0 and 1, but is able to outperform both alternative methods by the end of the third task training. This could be indicative of an ability to better maintain network performance when many new tasks are introduced. It is possible that the rigidity of the Laplace approximation and per-sample weight calculation means that as more tasks are introduced it becomes increasingly hard to find free weights to use, and thus if all weights are of equal importance, then weights of previous importance will be used more. The sampling approach of VAE may lead to generally worse performance compared to the EWC model, but it also appears to add robustness to maintain model performance on old tasks as a larger number of new tasks get added. This is supported not only by the median, but also the standard deviation of the performance towards the 80^{th} epoch.

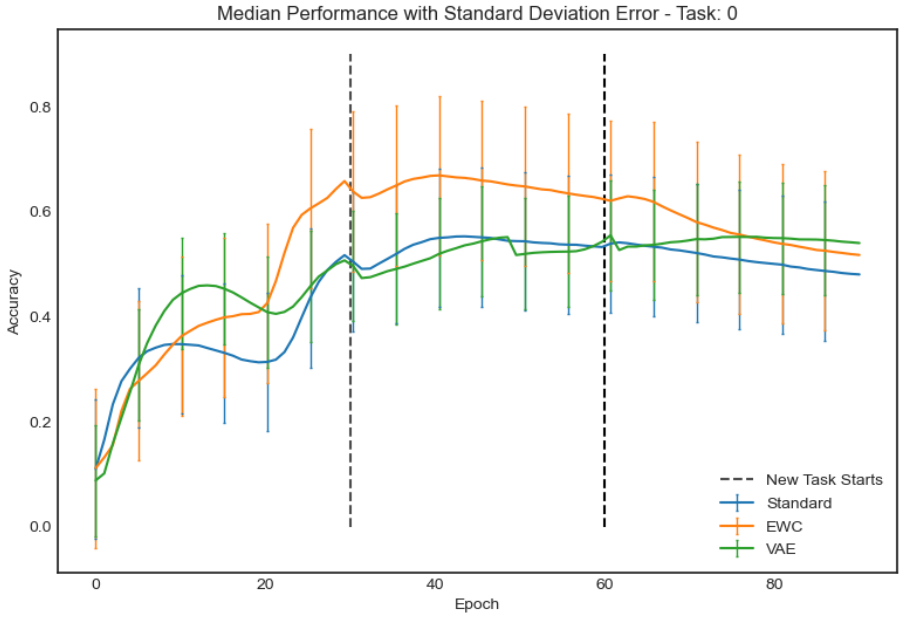

Figure 38: Task 0 performance of each model

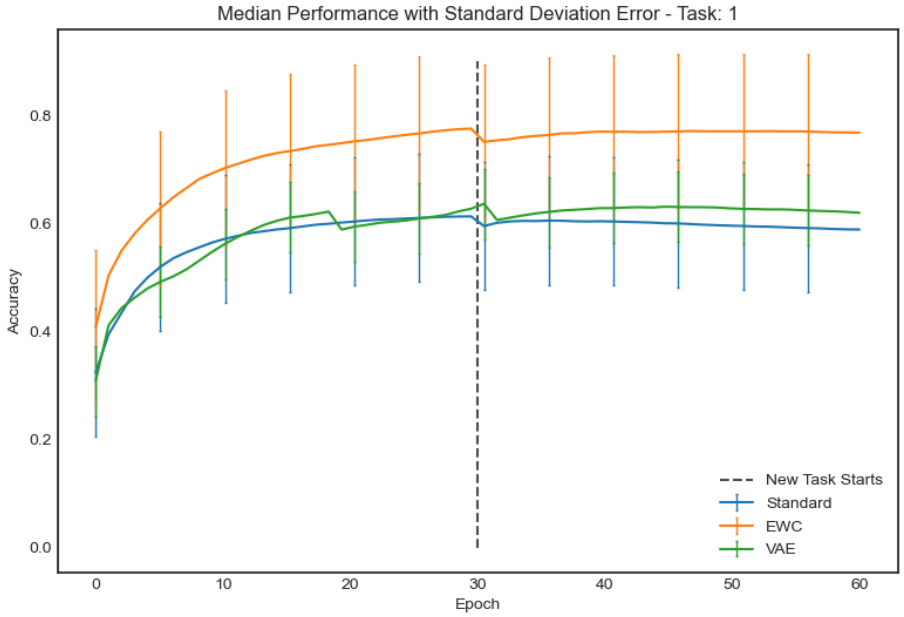

Figure 39: Task 1 performance of each model

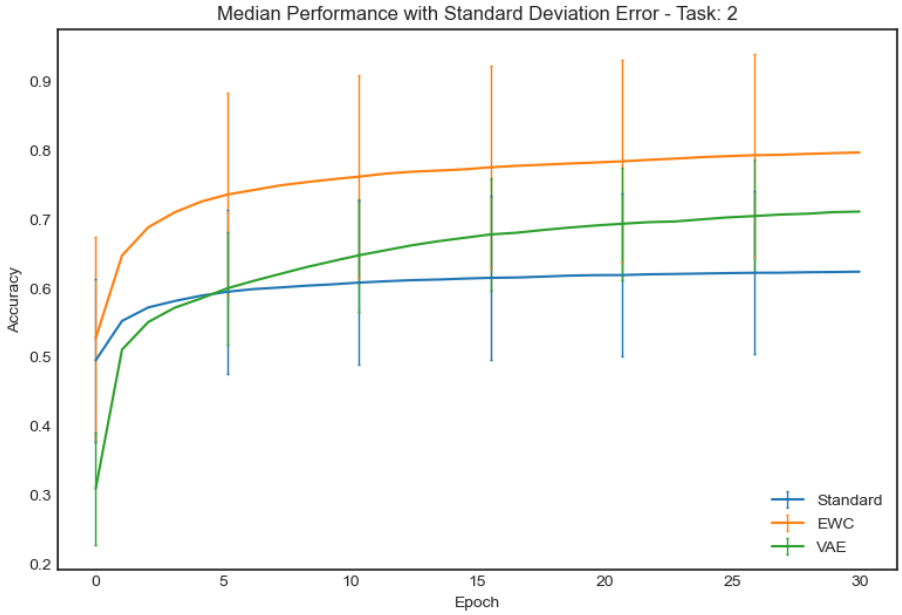

Figure 40: Task 2 performance of each model

While both the EWC and standard network improve rapidly before deteriorating as subsequent tasks are introduced, the VAE continuously improves throughout the entire 90 epochs. This could indicate several things: firstly, the introduction of the VAE may mean training for longer is necessary to realise the maximum performance of the model. Secondly, the sampling approach to weight constraints may be allowing the new areas of the parameter space to be explored, thus allowing improvements to occur during other tasks. Another observation on the behaviour of the VAE system is that it is very unstable. Unlike the smooth rates of change found in the other methods, the VAE displays almost discrete jumps in performance at certain epochs where performance will improve massively over a small number of epochs. Even stranger is that these epochs appear to be randomly located, not in areas where new tasks are introduced. This behaviour may support the notion that the explore-exploit trade-off of the sampling system is successfully finding new areas of the parameter space which leads to the periodic rapid improvement.

For the second task, EWC massively outperforms its counterparts, while the VAE system slightly outperforms the standard model once the third task is introduced. The ability of the EWC model to outperform the standard model on subsequent tasks is incongruent with the findings in the original paper on the same task (Kirkpatrick et al., 2017), where the EWC algorithm led to slightly reduced performance on later tasks due to the constraints imposed. The EWC

behaviour on the first two tasks indicates that the information retention system is working as intended, and that the over-performance is an unexpected side-effect. Furthermore, the variance of all three models was rather large. It is possible that, as the permutations are randomly created at run-time, different permutation sets are easier or harder to learn together than other permutation sets, and so the inter-run variance becomes very large.

For the final task, the order of performance is maintained with both EWC and VAE outperforming the standard model, but with EWC outperforming the VAE. As with task 0, the VAE is still improving at a significant rate at the termination of training (all three models appear to still be improving, but the VAE has the steepest gradient). This further supports the notion that the VAE model may need longer to train to reach its optimum performance.

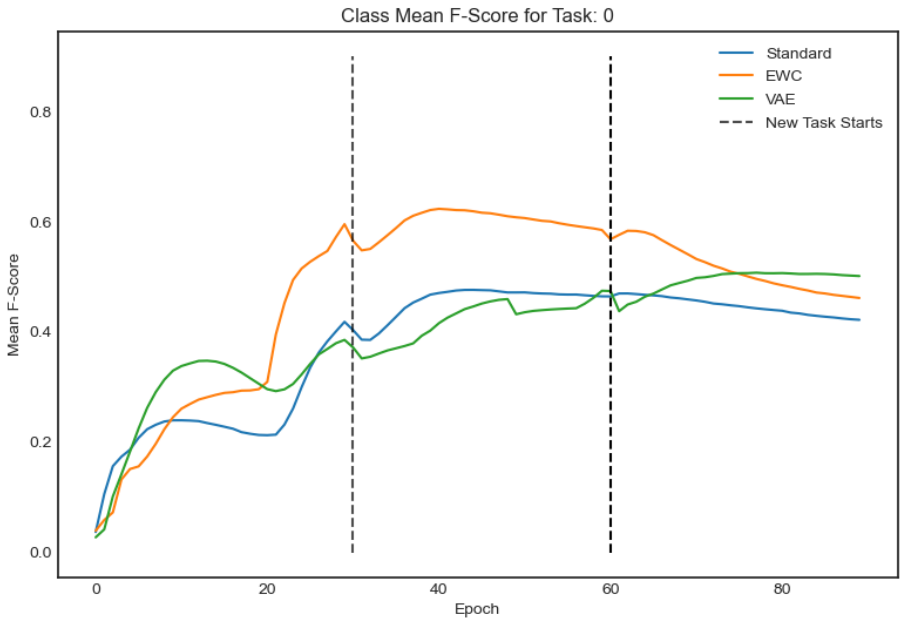

Figure 41: Mean F-score for the original MNIST dataset

The class mean F-score for all models across all tasks can be seen in figures 41, 42, 43. Due to the exploratory nature of this contribution, there is little that can be concluded from the initial findings. A series of Alexander Govern tests indicates that there is not a significant difference between the final model performances for all three tasks with p-values of 0.400, 0.81, and 0.54, respectively. While nothing statistically conclusive may be drawn from this experiment, a working implementation of the VAE sampling system has been successfully implemented, which empirically outperforms a standard network on the MNIST permutation task. Furthermore,

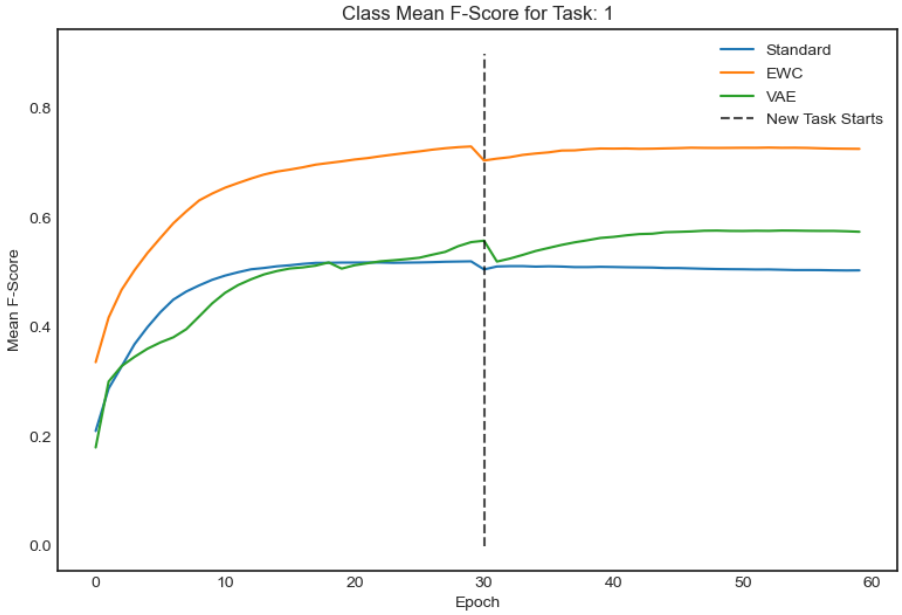

Figure 42: Mean F-score for the first permutation of MNIST

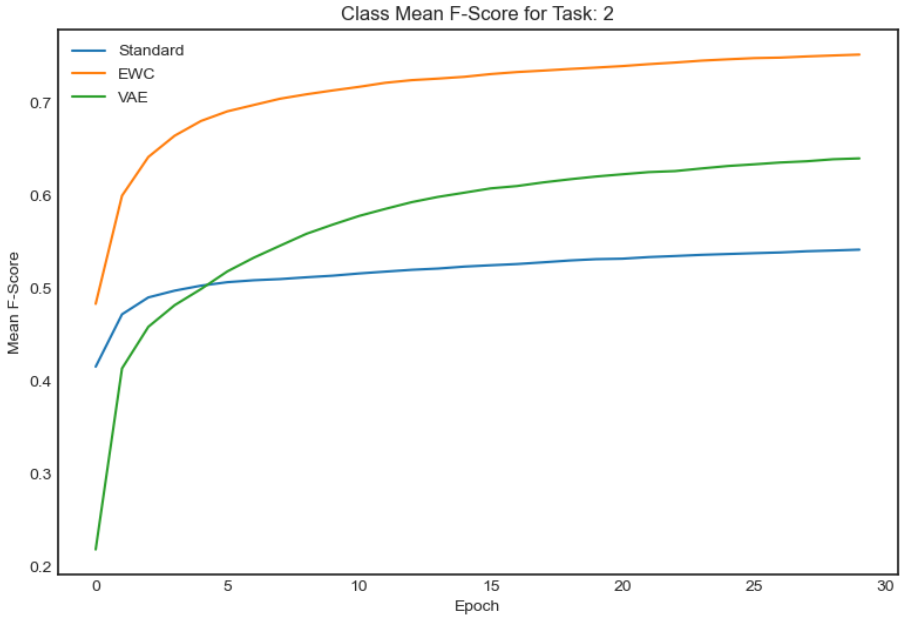

Figure 43: Mean F-score for the second permutation of MNIST

there is evidence to suggest that the approach offers alternative benefits that may not be present in the original EWC approach, despite original EWC outperforming the VAE approach in the experiments conducted here. Further study of the approach, along with more optimisation of the model, are necessary to truly conclude the efficacy of the strategy.

6 Discussion

While some of the findings of this work were not supported by statistical testing, they do offer empirical evidence to support interesting hypotheses, suggesting they warrant further research. To answer the first research question postulated, a time-series LSTM was able to successfully classify precipitation for the next 24 hours, given a set of sensor readings. An average accuracy of 72.35% is in practice a very good accuracy for a such a challenging task, particularly given the added complexity of handled massive class imbalance. Future work could explore alternative class imbalance strategies such as smote or GANs for data generation. While they were investigated briefly, a limitation is that due to the time-series nature of the data, it is not sufficient to generate a single data point, but rather a chain of values that also feed seamlessly into the existing data is necessary. This is non-trivial, and could constitute an entire research project on its own, however solving the problem could yield a massive improvement to the LSTM predictions.

Both of the LSTM models outperformed the human agent and soil-moisture sensor methods for soil-moisture optimisation, as both provided more yield from the same crop and maintained a better average soil-moisture level across the period. Thus, the future soil-moisture state does improve soil-moisture levels, supporting the second hypothesis that the future soil-moisture state is important for optimising the moisture content (albeit further experimentation is necessary to achieve the support of the statistical testing). Future work could also compare the performance of the models to a human expert, rather than a hobbyist human agent, to evaluate if the LSTM strategies can still outperform the human.

To answer the third research question, lifelong learning does alleviate the pressures of transferring a precipitation model to a new region, as the elastic weight consolidation network was statistically significantly better than the base LSTM when moving from the MeteoNet dataset to the NCAR dataset. Furthermore, the later stages of the real-world deployment do indicate that once stabilised, the EWC network is better than the base LSTM, although this is not yet supported by statistical testing.

While overall the base LSTM did outperform the closed-loop EWC system, there is evidence to suggest that this is because of poor hyper-parameter choice and a difficult initial adaptation period. If a lower learning rate was selected, and the EWC model was deployed for a period to manage the soil-moisture *before* the crop was planted, the acclimatisation period would be

less severe and would also occur before the crop was planted. Once the initial acclimatisation period had occurred, the network training was far smoother and the resultant model outperformed the base LSTM model. It is also possible that simply changing the learning rate would be sufficient, and early deployment would not be necessary. While there is evidence that the closed-loop control did improve the soil-moisture levels, particular at the end of the observation period, there is not sufficient evidence to convincingly claim that it definitively improves the soil-moisture optimisation.

The EWC algorithm outperformed the variational autoencoder approach over the 3 MNISt permutation tasks presented. However, there are several behaviours from the VAE model that are indicative of desirable behaviours, such as the ability to find otherwise forbidden solutions in the parameter space, or the ability to maintain performance on old tasks when a greater number of subsequent tasks are introduced. While it did not outperform the original EWC, the VAE strategy did outperform the base MLP, despite a relatively little amount of time being invested into creating the algorithm. With more time dedicated to optimising the approach, it may improve upon the original EWC methodology. Future work could be conducted on completing a wider grid search, on altering the sample rate of the weights, as well as on implementing variational Bayes' on a per-weight basis, to evaluate the impact of variational Bayes' against the impact of the subset sampling approach. Alternatively, there may be scope to extend the EWC algorithm to operate in a multi-agent system. Given a set of agents with the same underlying neural architecture, all of whom utilise lifelong learning to solve tasks, agents could learn from each other to solve new tasks by sharing weight sets between them, utilising EWC to preserve their current functionality while learning from other agents.

The sensor system has many areas to improve. Most notably, higher quality soil moisture sensors are required, but also a better approach to wiring and soldering is needed to prevent wires from interfering with each other as well as to avoid loose connections. Further hardware extensions could also be added to automate the process of watering the crop. Such systems already exist, and integrating them with the system here would create a full-stack watering system that could monitor the atmospheric and soil conditions, use an LSTM to inform watering regimens, before actually completing the watering process itself. This would completely remove the human from the loop. Furthermore, a multi-agent approach could be taken to deploy many of these systems across a large heterogeneous farm, to allow the system to manage an entire large scale farm environment.

7 Reflective Analysis

The project was of substantial size and complexity, comprised of several sub-sections that could have been expanded to an entire project on their own. After initial discussions took place to determine the general direction of the project, an agile approach was used to focus on achieving obtainable short-term goals via weekly sprints. Weekly sprints worked well, as regular supervisor meetings allowed consistent fine tuning of the project without loosing too much time redoing sections, as problems could be identified and alleviated before much time had been invested in the approach. These meetings were similar to gathering requirements from a client or product owner within agile methodology frameworks such as Scrum, although the links are tenuous as there were not enough participants in the project to allocate every Scrum role.

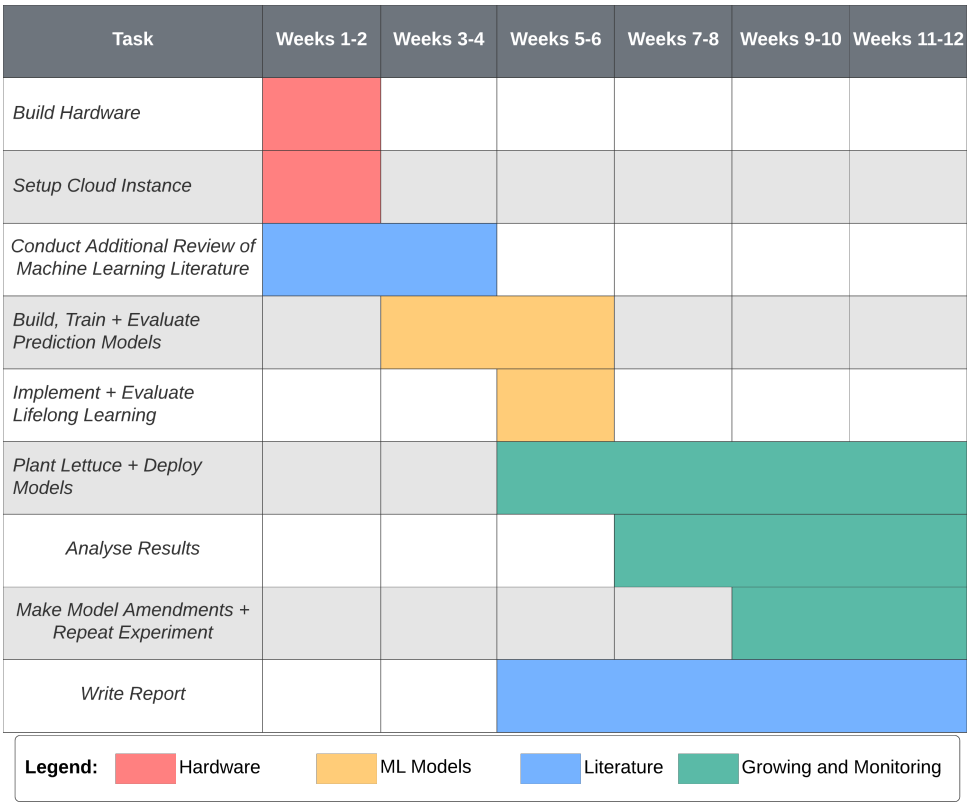

Figure 44: Gantt chart for project plan

Figure 44 shows a Gantt chart that was created at the start of the project. In practice, very little of the actual work structure aligned with the chart. The development of the machine learning model was the first task completed, as it was believed to be the most important task within the project, and thus extra emphasis was placed on achieving good performing models. The advantage of this was that when regression proved to be ineffective, there was plenty of

time to move to classification as the model development was started early. However, a problem with starting the hardware later was twofold: firstly, when 3D printing parts, there was a bottleneck where little could be done to advance the project further as the model development had already been conducted and, secondly, when the buck converter issue occurred which destroyed several components, there was not the time to wait for new components to arrive and so the hardware had to be deployed later than the radishes, leading to a reliance upon other data sources initially. Another problem that occurred was the final precipitation model relied upon historic data. As such, the planting of the radishes had to be delayed a week to collect this data. This was problematic as then the radishes had to be harvested slightly earlier than desired. The main issue with the time management of the project was that the project was just so large, containing four separate aspects (precipitation model, hardware, lifelong learning, VAE sampling), that there was no extra time to accommodate unforeseen circumstances. This problem was exacerbated by the nature of having to wait for several tasks to complete, such as waiting for networks to train, radishes to grow or for hardware to arrive or print. Any setback meant these waiting periods had to be duplicated so it wasn't simply a case of working later or putting extra man hours in to the task. It is also worth noting the hard deadline for deployment was about a month earlier than the actual project deadline, as the radishes needed time to grow.

Not all of the time management was poor, however. The project was completed on time and even contained the VAE sampling section, that was not originally planned or expected (it was a concept that developed part way through the project). Finding the time for this extra work allowed the project to contain a novel contribution to the field, and it completed during the weeks where the radishes were growing, meaning it was not delaying the production of other project sections.

Risk was managed in several ways during the project. To avoid data loss, GitHub was used to store backups of all work and results on the cloud. The version control aspects of Git ensured that any previous versions could be restored should any changes result in serious and irreversible errors. Contingencies were made in the event that the sensors failed by ensuring access to UK Met Office data was acquired as was a handheld soil moisture sensor. This guaranteed that even in the event that no hardware was deployed, the precipitation models could still be applied to the planted radishes. Finally, there was some risk that birds would damage or eat the radish crops once they sprouted. To alleviate this risk, netting was placed over the plant pots to dissuade the birds from reaching the crop.

The contributions made were both substantial, with the project producing: an effective precipitation prediction model (along with an evaluation of alternative but inferior models), an implementation of EWC with said model, custom made hardware for deployment, a WiFi-connected weather station, and a novel extension of the EWC algorithm. While the hardware did work, there is room for improvement, as the deployment was late and the soil moisture sensors were inaccurate. While the late deployment was avoidable, the poor soil moisture sensors were not, as this is simply a financial barrier. Taking just a small step up in quality of sensor could have avoided this, but this was unfortunately an issue not with the design or implementation of the system, but with the financial constraints and thus could not be rectified during the 3 month project. Overall, the project successfully answered the research questions it set out to address. While there is much room for improvement and scope for future work, the contributions of this work are substantial, novel, and may be deployed to solve important real-world problems.

8 Conclusion

This work explored several topics pertinent to the closed-loop control of soil-moisture, finding substantial evidence to support the hypotheses discussed in section 1. Neural network models were trained against a precipitation prediction dataset and critically evaluated before the best performing model, a time-series LSTM, was deployed in a real-world scenario to optimise soil-moisture levels for radishes. A bespoke weather station system was designed and built, capable of collecting atmospheric and soil-moisture data safely. The atmospheric data formed LSTM input vectors, however the soil-moisture sensors were inaccurate and were overshadowed by a handheld moisture sensor.

Elastic weight consolidation was used to facilitate the transferring of the precipitation prediction model from an existing dataset to real-world deployment in a different geographic region. Soil-moisture sensor data was subsequently used to provide feedback in a closed-loop control system, facilitating the training of the EWC LSTM. Both LSTM strategies outperformed alternative, contemporary methods, thus highlighting the efficacy of predicting future states when optimising soil-moisture levels.

Finally, a novel extension to the EWC algorithm was proposed, implemented, and critically evaluated. The VAE based per-subset sampling approach to weight consolidation was not statistically different to the EWC network, but empirically the evidence suggests that the approach is worse than the original strategy, but may be better for problems with a large number of tasks.

Future work can explore a wide-range of topics to extend the work presented here, such as improving the VAE strategy, expanding on the experimentation and real-world deployment of the LSTM and EWC-LSTM networks, improving the hardware weather station to facilitate autonomous watering, or even moving the system to accommodate a multi-agent approach. Further research must be conducted to conclude which extensions offer the greatest impact and are of most significance.

References

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y. and Zheng, X. (2015), 'TensorFlow: Large-scale machine learning on heterogeneous systems'. Software available from tensorflow.org.
URL: *http://tensorflow.org/*

Ambler-Edwards, S., Bailey, K. S., Kiff, A., Lang, T., Lee, R., Marsden, T. K., Simons, D. W. and Tibbs, H. (2009), 'Food futures: rethinking uk strategy. a chatham house report'.

Amini, A. and Soleimany, A. (n.d.), 'Deep generative modeling — mit 6.s191: Introduction to deep learning'.
URL: *IntroToDeepLearning.com*

Barthel, S. and Isendahl, C. (2013), 'Urban gardens, agriculture, and water management: Sources of resilience for long-term food security in cities', *Ecological economics* **86**, 224–234.

Barthel, S., Parker, J. and Ernstson, H. (2015), 'Food and green space in cities: A resilience lens on gardens and urban environmental movements', *Urban studies* **52**(7), 1321–1338.

Beddington, S. J. (2011), 'The future of food and farming', *International Journal of Agricultural Management* **1**(2), 2–6.

Blackmore, S. (1994), 'Precision farming: an introduction', *Outlook on agriculture* **23**(4), 275–280.

Blackmore, S. (2009), 'New concepts in agricultural automation'.

Brath, A., Burlando, P. and Rosso, R. (1988), Sensitivity analysis of real-time flood forecasting to on-line rainfall predictions, *in* 'Selected Papers from the Workshop on Natural Disasters in European-Mediterranean Countries, Perugia, Italy', pp. 469–488.

Choi, J., Yi, K. M., Kim, J., Choo, J., Kim, B., Chang, J.-Y., Gwon, Y. and Chang, H. J. (2020), 'Vab-al: Incorporating class imbalance and difficulty with variational bayes for active learning'.

Cover, T. and Hart, P. (1967), 'Nearest neighbor pattern classification', *IEEE transactions on information theory* **13**(1), 21–27.

Devi, S. R., Arulmozhivarman, P., Venkatesh, C. and Agarwal, P. (2016), 'Performance comparison of artificial neural network models for daily rainfall prediction', *International Journal of Automation and computing* **13**(5), 417–427.

Duckett, T., Pearson, S., Blackmore, S., Grieve, B., Chen, W.-H., Cielniak, G., Cleaversmith, J., Dai, J., Davis, S., Fox, C., From, P., Georgilas, I., Gill, R., Gould, I., Hanheide, M., Hunter, A., Iida, F., Mihalyova, L., Nefti-Meziani, S., Neumann, G., Paoletti, P., Pridmore, T., Ross, D., Smith, M., Stoelen, M., Swainson, M., Wane, S., Wilson, P., Wright, I. and Yang, G.-Z. (2018), 'Agricultural robotics: The future of robotic agriculture'.

Eller, H. and Denoth, A. (1996), 'A capacitive soil moisture sensor', *Journal of Hydrology* **185**(1-4), 137–146.

Elman, J. L. (1990), 'Finding structure in time', *Cognitive science* **14**(2), 179–211.

Farquhar, S. and Gal, Y. (2019), 'Towards robust evaluations of continual learning'.

French, M. N., Krajewski, W. F. and Cuykendall, R. R. (1992), 'Rainfall forecasting in space and time using a neural network', *Journal of hydrology* **137**(1-4), 1–31.

Goodfellow, I. J., Mirza, M., Xiao, D., Courville, A. and Bengio, Y. (2013), 'An empirical investigation of catastrophic forgetting in gradient-based neural networks', *arXiv preprint arXiv:1312.6211* .

Haidar, A. and Verma, B. (2018), 'Monthly rainfall forecasting using one-dimensional deep convolutional neural network', *IEEE Access* **6**, 69053–69063.

Hassani, H. (2007), 'Singular spectrum analysis: methodology and comparison'.

Hataya, R. (2018), 'Ewc.pytorch', `https://github.com/moskomule/ewc.pytorch`.

Hochreiter, S. and Schmidhuber, J. (1997), 'Long short-term memory', *Neural computation* **9**(8), 1735–1780.

Kingma, D. P. and Welling, M. (2013), 'Auto-encoding variational bayes'.

Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A. A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A. et al. (2017), 'Overcoming catastrophic forgetting in neural networks', *Proceedings of the national academy of sciences* **114**(13), 3521–3526.

Krizhevsky, A., Sutskever, I. and Hinton, G. E. (2012), Imagenet classification with deep convolutional neural networks, *in* 'Advances in neural information processing systems', pp. 1097–1105.

Kullback, S. and Leibler, R. A. (1951), 'On information and sufficiency', *The annals of mathematical statistics* **22**(1), 79–86.

Larvor, G., Berthomier, L., Chabot, V., Pape, B. L., Pradel, B. and Perez, L. (2020), 'Meteonet, an open reference weather dataset by meteo france', *www.kaggle.com/katerpillar/meteonet* .

LeCun, Y., Bengio, Y. and Hinton, G. (2015), 'Deep learning', *nature* **521**(7553), 436–444.

LeCun, Y., Bottou, L., Bengio, Y. and Haffner, P. (1998), 'Gradient-based learning applied to document recognition', *Proceedings of the IEEE* **86**(11), 2278–2324.

Li, Z. and Hoiem, D. (2017), 'Learning without forgetting', *IEEE transactions on pattern analysis and machine intelligence* **40**(12), 2935–2947.

Liu, X., IJzerman, A. and Westen, G. (2020), *Computational Approaches for De Novo Drug Design: Past, Present, and Future*, Vol. 2190, pp. 139–165.

López-granados, F. (2011), 'Weed detection for site-specific weed management: mapping and real-time approaches', *Weed Research* **51**(1), 1–11.

Luk, K. C., Ball, J. and Sharma, A. (2001), 'An application of artificial neural networks for rainfall forecasting', *Mathematical and Computer Modelling* **33**(6), 683–693.
URL: *https://www.sciencedirect.com/science/article/pii/S0895717700002727*

McCloskey, M. and Cohen, N. J. (1989), Catastrophic interference in connectionist networks: The sequential learning problem, *in* 'Psychology of learning and motivation', Vol. 24, Elsevier, pp. 109–165.

Mougeot, L. J. et al. (2000), 'Urban agriculture: definition, presence, potentials and risks', *Growing cities, growing food: Urban agriculture on the policy agenda* **1**, 42.

O'Neal, M. R., Frankenberger, J. R., Ess, D. R. et al. (2000), Spatial precipitation variability in the choice of nitrogen fertilization rates., *in* 'Proceedings of the 5th International Conference on Precision Agriculture, Bloomington, Minnesota, USA, 16-19 July, 2000', American Society of Agronomy, pp. 1–15.

Parisi, G. I., Kemker, R., Part, J. L., Kanan, C. and Wermter, S. (2019), 'Continual lifelong learning with neural networks: A review', *Neural Networks* **113**, 54–71.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J. and Chintala, S. (2019), Pytorch: An imperative style, high-performance deep learning library, *in* H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox and R. Garnett, eds, 'Advances in Neural Information Processing Systems 32', Curran Associates, Inc., pp. 8024–8035.
URL: *http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf*

Pawitan, Y. (2001), *In all likelihood: statistical modelling and inference using likelihood*, Oxford University Press.

Pierce, F. J. and Nowak, P. (1999), 'Aspects of precision agriculture', *Advances in agronomy* **67**, 1–85.

Poornima, S. and Pushpalatha, M. (2019), 'Prediction of rainfall using intensified lstm based recurrent neural network with weighted linear units', *Atmosphere* **10**(11), 668.

Qing, X. and Niu, Y. (2018), 'Hourly day-ahead solar irradiance prediction using weather forecasts by lstm', *Energy* **148**, 461–468.

Qiu, M., Zhao, P., Zhang, K., Huang, J., Shi, X., Wang, X. and Chu, W. (2017), A short-term rainfall prediction model using multi-task convolutional neural networks, *in* '2017 IEEE International Conference on Data Mining (ICDM)', pp. 395–404.

Robins, A. (1993), Catastrophic forgetting in neural networks: the role of rehearsal mechanisms, *in* 'Proceedings 1993 The First New Zealand International Two-Stream Conference on Artificial Neural Networks and Expert Systems', IEEE, pp. 65–68.

Rumelhart, D. E., Hinton, G. E. and Williams, R. J. (1985), Learning internal representations by error propagation, Technical report, California Univ San Diego La Jolla Inst for Cognitive Science.

Rusu, A. A., Rabinowitz, N. C., Desjardins, G., Soyer, H., Kirkpatrick, J., Kavukcuoglu, K., Pascanu, R. and Hadsell, R. (2016), 'Progressive neural networks', *arXiv preprint arXiv:1606.04671* .

Sharma, K. (2021), 'National center for atmospheric research dataset for rainfall predictions'.

Toth, E., Brath, A. and Montanari, A. (2000), 'Comparison of short-term rainfall prediction models for real-time flood forecasting', *Journal of hydrology* **239**(1-4), 132–147.

Van Rossum, G. and Drake, F. L. (2009), *Python 3 Reference Manual*, CreateSpace, Scotts Valley, CA.

Veksler, O. (2004), 'Cs434a/541a: Pattern recognition'.

Wu, C., Chau, K. W. and Fan, C. (2010), 'Prediction of rainfall time series using modular artificial neural networks coupled with data-preprocessing techniques', *Journal of Hydrology* **389**(1-2), 146–167.

Zezza, A. and Tasciotti, L. (2008), Does urban agriculture enhance dietary diversity? empirical evidence from a sample of developing countries, Technical report.

Zhang, N., Wang, M. and Wang, N. (2002), 'Precision agriculture—a worldwide overview', *Computers and Electronics in Agriculture* **36**(2), 113–132.
URL: *https://www.sciencedirect.com/science/article/pii/S0168169902000960*