A Gradient Descent Multi-Algorithm Grid Search Optimization of Deep Learning for Sensor Fusion

Thomas M. Booth
309th SWEG/EDDGE Team
U.S. Air Force
Hill AFB, UT, USA
thomas.booth.2@us.af.mil
tom.booth@NexusDE.com

Sudipto Ghosh
Senior Member, IEEE
Department of Computer Science
Colorado State University
Fort Collins, CO, USA
sudipto.ghosh@colostate.edu

***Abstract*—Sensor fusion approaches combine data from a suite of sensors into an integrated solution that represents the target environment more accurately than that produced by an individual sensor. Deep learning (DL) based approaches can address challenges with sensor fusion more accurately than classical approaches. However, the accuracy of the selected approach can change when sensors are modified, upgraded or swapped out within the system of sensors. Historically, this can require an expensive manual refactor of the sensor fusion solution.**

This paper develops 12 DL-based sensor fusion approaches and proposes a systematic and iterative methodology for selecting an optimal DL approach and hyperparameter settings simultaneously. The Gradient Descent Multi-Algorithm Grid Search (GD-MAGS) methodology is an iterative grid search technique enhanced by gradient descent predictions and expanded to exchange performance measure information across concurrently running DL-based approaches. Additionally, at each iteration, the worst two performing DL approaches are pruned to reduce the resource usage as computational expense increases from hyperparameter tuning. We evaluate this methodology using an open source, time-series aircraft data set trained on the aircraft's altitude using multi-modal sensors that measure variables such as velocities, accelerations, pressures, temperatures, and aircraft orientation and position. We demonstrate the selection of an optimal DL model and an increase of 88% in model accuracy compared to the other 11 DL approaches analyzed. Verification of the model selected shows that it outperforms pruned models on data from other aircraft with the same system of sensors.

***Index Terms*—Deep Learning, Sensor Fusion, Optimization**

I. Introduction

Multi-sensor systems such as aircraft and autonomous vehicles use a suite of multi-modal sensors to collect data about their environment. A sensor fusion approach combines the sensor data into an integrated solution that represents the environment more accurately than what could be accomplished using a single sensor. For example, in this paper, variables such as velocities, accelerations, pressures, temperatures, and aircraft orientation and position are fused to calculate the aircraft's altitude. In practice, certified altimeters reliably perform this task. However, altitude was selected as a certified truth source target to demonstrate the Gradient Descent Multi-Algorithm Grid Search (GD-MAGS) methodology.

Multi-modal sensor fusion presents numerous challenges that arise from the data to be fused, the imperfection and diversity of the sensor technologies, and the nature of the environment. Of the many challenges listed in Khaleghi et al. [1], this paper addresses the following: outliers and spurious data, conflicting data, data imperfection, data modality, operational timing, and static vs dynamic phenomenon.

Deep Learning (DL) is quickly becoming the leading approach compared to previous classical methods in time-series prediction algorithms as shown by the performance of Recurrent Neural Network (RNN) based approaches in the M5 competition [2]. Therefore, this paper developed 12 DL-based approaches to sensor fusion, for the target data set, derived from a combination of techniques to common problems seen in this data set and mentioned in Pires et al. [3]. These approaches were created from a baseline RNN algorithm variant called Long-Short Term Memory (LSTM) and 11 other combinations of data ingest and batch mode techniques. We explore the advantages of applying a supplemental data modeling technique which ingests additional data for situations where there is insufficient data to properly train RNN models. Since data filtering techniques are common prior to training models with empirical data, we apply two such techniques on three noisy signals providing unexpected and conflicting values. We also apply a classification technique since data classification can lead to more accurate regressive DL models. Additionally, two batch mode techniques are implemented to identify the differences between the two.

For developing and deploying sensor fusion solutions, engineers need a resource efficient and systematic methodology to select the most accurate DL approach with optimal hyperparameter settings. Training and testing a combination of DL approaches and LSTM hyperparameters on a statistically significant and representative subset of the data can reduce time and resource usage. From here, it follows that the selected model will likely perform well when trained and tested on the full data set.

This paper takes a first step in addressing this problem by providing an automatable and systematic methodology for selecting an optimal DL approach and RNN hyperparameters for a set of data from a specific sensor configuration. The next steps in this process will involve standardizing DL model performance data import into a Model Based System

Engineering (MBSE) SysML model for data driven decisions and resource limited sensor architecture optimizations.

The GD-MAGS methodology is a modification of the iterative grid search [4] technique. However, it expands on it by exchanging Neural Network (NN) performance information across multiple NN approaches rather than a single NN. Additionally, this information is used with gradient descent based equations to speed up convergence across multiple DL approaches. This paper evaluates the potential accuracy and flexibility of a RNN, specifically the LSTM algorithm, applied to real-time aircraft data from an open NASA data set [5]. The NASA data was recorded onboard a single type of regional jet operating in commercial service over a three year period and was chosen for its availability to publish and the known accuracy in the altitude measurement. The LSTM algorithm is trained using inputs from the aircraft multi-modal sensors with the altitude as the target variable. We evaluated the accuracy of this selection methodology on two other aircraft of the same type with the same sensor suite to verify the selected DL model is an optimal solution for those as well.

Section II summarizes previous work on sensor data fusion to orient the reader to the typical difficulties involved. Section III describes the NASA data set. We describe the 12 DL-based approaches and the GD-MAGS methodology in Section IV. Then we present the results and discuss them in Sections V and VI, respectively. Lastly, we present our conclusions and outline directions for future work in Section VII.

II. Related Work

This section summarizes previous sensor fusion and related time-series prediction work as of Aug 2022 using DL approaches, each with advantages and disadvantages. These DL approaches demonstrate valuable results for time-series forecasting models and for real-time sensor fusion applications. While the forecasting models employ hyperparameter tuning and ensemble approach optimizations, it is unclear that any of the sensor fusion applications apply tuning or optimizations across multiple DL approaches or on their respective hyperparameters for resource limited embedded systems typically found with sensor systems.

The area of DL research has seen significant growth in popularity and application in recent years [2]. Recently, the data forecasting community has shown that DL-based approaches or a hybrid classical and DL approach can be more accurate compared to previous classical methods alone in time-series prediction algorithms [2]. N-BEATS [6], DeepAR [7], and Temporal Fusion Transformers for Interpretable Multi-Horizon Time Series Forecasting [8] are a few of the state-of-the-art approaches in forecasting models for accuracy, speed of training and speed of prediction with a trained model. However, these large, time-series forecasting models, such as DeepAR [7], take on the order of tens of minutes for a prediction with a pre-trained model. This prediction latency does not meet the timing requirements needed for real-time sensor fusion algorithms on embedded systems which can only tolerate latency on the order of milliseconds depending on the application. The two main DL techniques used in sensor fusion are Convolutional Neural Networks (CNN) and RNNs [4].

Hyperparameter tuning for RNN, CNN or any other ML algorithm is vital to ensure the true performance is reached [9], [10]. However, the methodology described by Weerts et al. [9] requires on the order of 1,000 hyperparameter configurations trained per data set and ML algorithm which is a large computational cost. Additionally, Oreshkin et al. [6] emphasises the importance of optimizing the ensemble approaches to time-series forecasting. These best practices in time-series forecasting appear to be missing in many DL-based sensor fusion papers.

Two commonly used methods of hyperparameter tuning is grid search and random search, where grid search is common practice with three or fewer hyperparameters [4]. Bergstra and Bengio [11] state that random search hyperparameter optimization is more efficient than grid search for solutions that are not sensitive to the hyperparameters being adjusted. For the application in this paper, the hyperparameters being tuned show that they strongly affect the performance measure, therefore the grid search technique was the basis of our gradient descent implementation.

In terms of sensor fusion applications, several papers [7], [12], [13] implemented some form of DL algorithms with success, however, it was unclear if any method was used to find an optimized combination of ensemble layer configurations, data ingest approaches and hyperparameters combined. To the best of our knowledge, our optimal selection methodology for deep learning approaches enhances the state of the art for DL sensor data fusion.

III. NASA Aircraft Data Set

We use a publicly available NASA data set [5] to develop, demonstrate, and evaluate the methodology. The data was recorded onboard multiple aircraft with the same sensor suite of a single type of regional jet operating in commercial service. The data files contain detailed aircraft dynamics, system performance and other engineering parameters captured on the data recorder.

The data set contains information for 35 different aircraft tail numbers averaging 5,355 flight files per aircraft with an average file size of 2MB. Each flight file contains an average of 125,000 sample points and represents a full flight that includes aircraft ground operations, take-off, flight, landing, and post landing ground operations. Additionally, each file has 188 aircraft time-series variables that include sensor data and other serial bus messages. These messages varied in recorded frequency between 0.25 and 16 Hz. Using prior aircraft knowledge, we manually identified 26 variables as DL model inputs with an altitude target variable for a total of 27 variables. Of these 26 input variables there are 8 different data modalities. The sensors on this aircraft collected heterogeneous data such as altitude, altitude rate, acceleration, pressure, temperature, speed, orientation, and position. While several chosen variables, such as Selected Airspeed, Selected Vertical Speed, and Ground Speed may not be required for

the DL approaches to accurately model altitude, we included them to mimic some superfluous and possibly conflicting data.

TABLE I
SENSOR VARIABLE NAMES AND DESCRIPTIONS

Name	Units	Rate(Hz)	Description
ALT	feet	4	Pressure altitude LSP **(Truth Source)**
Time	secs	16	Time in GMT seconds
FPAC	G	16	Flight path acceleration
BLAC	G	16	Body longitudinal acceleration
CTAC	G	16	Cross track acceleration
VRTG	G	8	Vertical acceleration
LATG	G	4	Lateral acceleration
LONG	G	4	Longitudinal acceleration
RALT	feet	8	Radio altitude LSP
ALTR	ft/min	4	Altitude rate
IVV	ft/min	16	Inertial vertical speed
VSPS	ft/min	1	Selected vertical speed
PSA	mbars	2	Average static pressure
PI	mbars	2	Impact (dynamic) pressure
PT	mbars	2	Total pressure
TAS	knots	4	True airspeed
CAS	knots	4	Calculated airspeed
GS	knots	4	Ground speed
WS	knots	4	Wind speed
CASS	knots	1	Selected airspeed
PTCH	deg	8	Aircraft pitch angle
ROLL	deg	8	Aircraft roll angle
DA	deg	4	Aircraft drift angle
TAT	°C	1	Total air temp
SAT	°C	1	Static air temp
LATP	deg	1	Latitude position LSP
LONP	deg	1	Longitude position LSP

Table I shows the 26 input variables and the target `ALT` variable, but also includes units, recorded frequencies, and brief descriptions provided by the NASA source files. We created continuous data at 16 Hz from signals recorded at lower rates using the pad method from Python `pandas` as a requirement for input into the LSTM algorithm.

Fig. 1 shows a single flight with 11 of the 26 variables aligned and plotted against `Time` to provide insight into a few of the challenges with fusing multi-modal sensor data. For instance Fig. 1(a) shows the aircraft's ground operations as a flat line of `ALT` before take-off and after landing. However, Figs. 1(b) and 1(d) show large fluctuations in altitude rate (`ALTR`) and body long acceleration (`BLAC`), respectively, while on the ground which is in conflict with `ALT`. Additionally, Fig. 1(c) shows unrealistic periodic data supplied from vertical, lateral, and longitudinal accelerations (`VRTG, LATG, LONG`). The top and bottom red colored data in this subplot both belong to `VRTG` data. Realistic `VRTG` values for passenger aircraft are centered around 1 G so the periodic data shown near -3 G's are considered noise. The `LATG` and `LONG` data in this same subplot overlap significantly and have realistic values for passenger aircraft centered around 0 G's. However, they both exhibit similarly noisy behavior with unrealistic and periodic data at -1 G. These challenges in outliers and spurious data, conflicting data, and data imperfection mentioned above as well as data modality operational timing, and static vs dynamic phenomenon drove

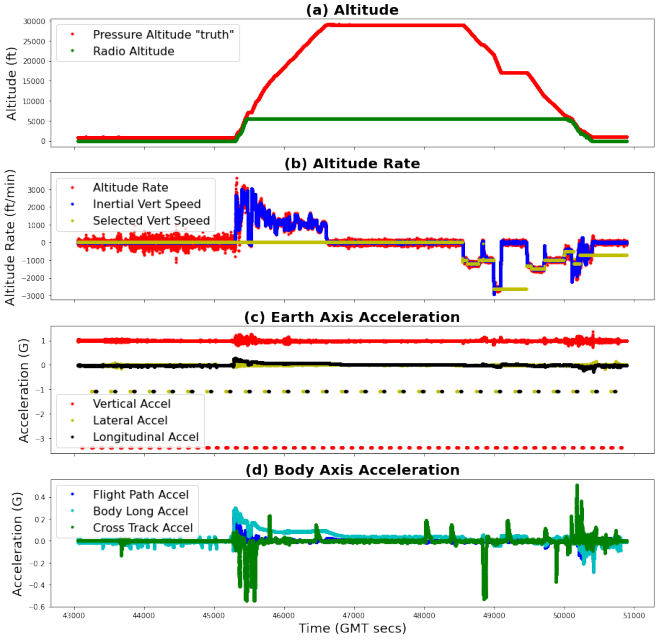

Fig. 1. Aircraft altitude, altitude rate and acceleration

the need to develop the 12 different DL approaches described in Section IV-A that are part of this paper's selection process.

IV. METHODOLOGY

This section describes the GD-MAGS methodology which enhances and adds multiple concurrent DL approaches to the grid search hyperparameter optimization technique. We describe the 12 DL-based approaches developed for the NASA data set, the initial LSTM hyperparameters, the rule that prunes the worst performing approaches, and the gradient descent rules used with grid search to tune the hyperparameters of the sensor fusion model. We use a portion of the NASA data set to reduce the computational resources and time of this study. One out of the 35 tail numbers from the data set was chosen at random along with 20 random flight files out of 5,355 of this tail number's data set. This subset of 20 files is used throughout this study to reduce stochastic results.

Fig. 2 is a Systems Modeling Language (SysML) internal block diagram (ibd) of the GD-MAGS optimization architecture. The `DL_Training` application shown creates the sensor fusion model using the `LSTM` algorithm with the `Adam` optimizer from the `PyTorch.NN` module. `DL_Training` trains each NN model using input training data, a DL approach, and hyperparameters provided by the `Bootstrap` application. `DL_Training` measures the model accuracy by comparing its output to the target altitude in the testing data using `PyTorch MSELoss` to calculate the performance measure, Mean Squared Error (MSE). The MSE function is based on (1), where n is the number of data points, and T_t and Y_t are the target and predicted values at time (t).

$$MSE = \frac{1}{n}\sum_{t=1}^{n}(T_t - Y_t)^2 \quad (1)$$

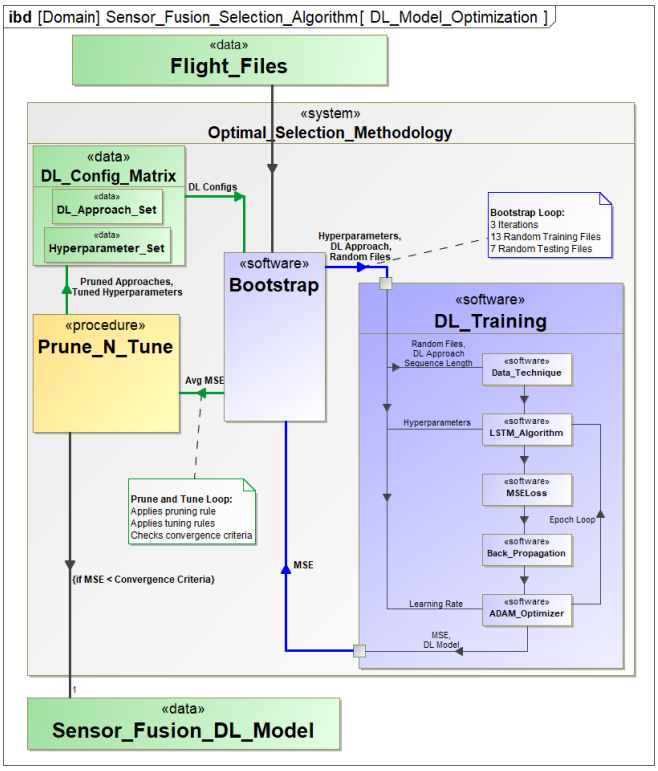

Fig. 2. SysML diagram of Optimal Selection Methodology for sensor fusion.

The prune and tune loop shown with green item flows in Fig. 2 contains the *Prune_N_Tune* procedure which includes the GD-MAGS optimization rules. The methodology starts with an initial DL Configuration matrix that contains the DL approaches and initial hyperparameters. *Prune_N_Tune* starts the first iteration by sending this initial DL configuration matrix to `Bootstrap` which completes 3 iterations of the bootstrap loop, shown with blue items flows, to calculate an average MSE. During each iteration, `Bootstrap` provides `DL_Training` 13 random training files, 7 random test files and a single DL configuration to produce a single model and MSE measurement. `Bootstrap` runs all configurations in the DL configuration matrix in parallel, after which, it returns the average model MSE for each configuration to *Prune_N_Tune*. Calculating the average MSE reduces stochastic data from affecting the selection results. Before starting a new refinement iteration, the minimum MSE of the DL configuration matrix is checked against the user specified convergence criterion. The refinement iterations will continue until this convergence criterion has been met or the optimal solution has been found.

The criterion can be an absolute MSE value, a gradient/optimality condition, or one of the user's choosing. When the criterion is satisfied, `Prune_N_Tune` will stop the refinement iterations and export the selected best performing model embedded with a Model Card [14] containing performance metrics and all information needed to reproduce it. If the convergence criterion isn't satisfied, a new refinement iteration begins and *Prune_N_Tune* applies the pruning and hyperparameter tuning rules based on the average MSE gradient objective function. *Prune_N_Tune* creates a new DL configuration matrix comprised of the pruned set of approaches and the newly tuned set of hyperparameters and sends this to `Bootstrap` which repeats the refinement iterations within the selection methodology until convergence is reached.

A. DL approaches

This section describes how each DL approach was created from a baseline LSTM approach. The GD-MAGS methodology performs best on a large set of DL approach combinations to better identify the optimal model for the data set. To build the other 11 DL approaches from the baseline LSTM approach several techniques were applied which include: two data filter techniques, one supplemental data model, one classification technique, and two batch mode techniques. Table II lists the names of these approaches that are compiled from the combination of techniques used to build each one. Also listed are the number of sensor variable inputs used for the LSTM algorithm. The fourth column is described and referenced in a later section.

TABLE II
DEEP LEARNING APPROACHES

#	Approach Name	Input #	Pruned$_i$
1	Baseline	26	5
2	Base+Outlier	26	5
3	Base+RemoveVar	23	Selected
4	Base+Outlier+Model	27	4
5	Base+RemoveVar+Model	24	4
6	Base+Class	26	4
7	Base+Class+Outlier	26	3
8	Base+Class+RemoveVar	23	4
9	Base+Class+Outlier+Model	27	3
10	Base+Class+RemoveVar+Model	24	4
11	Base+Flightbatch	26	2
12	Base+Outlier+Flightbatch	26	2

The *Baseline*, or *Base*, approach name or prefix refers to the baseline DL-based approach that implements the LSTM algorithm without additional techniques. This approach sets a frame of reference for comparison with the 11 other approaches that build on this baseline with other techniques. The following paragraphs describe the implementation of the six techniques used. These techniques are applied to the data in each approach in the order they appear in the approach names in Table II. These techniques are applied to the data before it is fed to the LSTM algorithm. The *Flightbatch* technique is an exception and is applied during the LSTM algorithm.

The *Outlier* technique is the first of two data filter techniques applied in this paper. It uses the Python Scikit-Learn `IsolationForest` outlier algorithm chosen for its performance against other well known outlier detection algorithms for data sets larger than 1,000 samples [15]. The `IsolationForest` algorithm was used to evaluate its effect on the model's accuracy by reducing outliers, or *noise*, in the three noisy signals discussed in Section III.

RemoveVar is the second data filter technique applied which removes the three noisy variables (`VRTG`, `LATG`, `LONG`)

from the input, thereby bypassing the need for outlier detection all together. The goal is to compare its model performance against the Isolation Forest algorithm performance.

The *Model* technique addresses possible limitations of the aircraft's sensors and available data recorded. This is a supplemental data model that provides ground elevation data as input to the LSTM algorithm. The intent is to provide additional data to the LSTM algorithm to more accurately model the target altitude (`ALT`) through a Digital Terrain Elevation Data (DTED) table lookup from the GMTED2010 [16] database. This database is commonly used on aircraft or can be added to the aircraft's software if needed.

The *Class* technique classifies the flight data as either ground operations or in-flight data to remove conflicting ground data from the training set. This technique deletes all ground data before the LSTM algorithm is trained on the rest of the sensor data, towards improving the final solution.

Two different batch mode techniques are included to analyze the effects of splitting time series data in two ways when training models. The first batch method, which is implemented in the baseline approach, appends the flight data from all 13 training flight files into a single data frame and trains the model in equal batch sizes of 50,000 data points at a time. This technique ignores the beginning and end of individual flight files and iteratively trains forward through the model, calculates loss, back-propagates the loss, and performs the single optimization step for each batch file until moving onto the next epoch. The second batch mode approach, known as *Flightbatch*, customizes each batch size to contain each of the 13 flight files and performs the same back-propagation steps as the baseline which is also describe in Yao et al. [13].

For each approach, the techniques above are applied to the input data, then the data is scaled near unity using scikit-learn `MinMaxScaler` and `fit_transform` to reduce truncation and rounding errors. Next, a sequence length sized windowing scheme is applied. Finally, the training and testing data partitions are created and the model is trained. The testing partition is scaled with the training `MinMaxScalar` values with scikit-learn's `transform` function before measuring MSE.

B. Initial LSTM hyperparameters

Initial learning rates (LR) and numbers of epochs (Epochs) are chosen for quick model convergence, then tuned in subsequent iterations. The purpose of this technique is to eliminate relatively poor performing approaches early in the process when it is computationally cheap. Initial values for number of hidden units (Units) were spread out evenly to produce an initial MSE gradient to inform the tuning rules. Sequence lengths 8 and 16 were chosen as factors of the 16 Hz signals. Selection of the RNN hyperparameters, guidance for initial values, and the information about the Adam optimization algorithm are not described here, but an introduction to these can be found in [4].

C. GD-MAGS Optimization Rules

This section describes the DL approach pruning rule and the gradient descent hyperparameter tuning rules applied to the grid search method across concurrent DL approaches.

Approach pruning rule: For each refinement iteration, remove the two worst performing DL approaches from the DL configuration matrix. Here, performance is based on the lowest MSE of best hyperparameters for the approach. More than two approaches can be removed per iteration if there is a consistent trend of poor approach performance across iterations. Be cautious when removing additional approaches before iteration three in order to have sufficient confidence.

Learning rate tuning rule: The MSE gradient from the best and worst performing hyperparameter configurations is the basis for the learning rate and epoch number tuning rules. The learning rate is reduced proportionally to this gradient with a diminishing damping factor applied. The factor slows the learning rate reduction early in the process when the gradient is large. However, this factor is reduced at each iteration as the solutions converge. The new learning rate for the next iteration (LR_{i+1}) is calculated using (2),

$$LR_{i+1} = \frac{LR_{i_{best}}}{ratio_{LR}} \tag{2}$$

where $LR_{i_{best}}$ is the best performing hyperparameter configuration learning rate for the current iteration i. The $ratio_{LR}$ is the learning rate ratio calculated in (3),

$$ratio_{LR} = \frac{1}{F_d}\frac{MSE_{worst}}{MSE_{best}} \tag{3}$$

where F_d is the diminishing damping factor, MSE_{worst} and MSE_{best} are the worst and best performing hyperparameter configuration's average MSE, respectively. The ratio $\frac{F_d MSE_{worst}}{MSE_{best}}$ must always remain greater than one and initial recommended values of F_d should produce LR_{i+1} less than 1 order of magnitude (OOM) smaller than LR_i. Very large initial F_d will drop the LR_{i+1} several OOM leading to convergence that will be overly resource intensive and the methodology will most likely skip optimum solutions. Very small F_D will increase LR_{i+1} compared to LR_i and lead to divergent solutions.

Epoch number tuning rule: Assuming the previous models were sufficiently converged, the number of epochs for the next iteration ($Epochs_{i+1}$) are adjusted using (4),

$$Epochs_{i+1} = Epochs_{i_{best}} \left(\frac{LR_{i_{best}}}{LR_{i+1}}) \right) \tag{4}$$

where $Epochs_{i_{best}}$ is the Epoch number of the best performing hyperparameter configuration. If there is indication this hyperparameter configuration sufficiently converged at a lower Epoch number, then reduce $Epochs_{i_{best}}$ to the Epoch number at convergence before calculating $Epochs_{i+1}$.

Hidden unit tuning rule: Hidden unit tuning can be difficult to implement and performance of the hidden units can change between iterations. Therefore it is recommended to keep a range of hidden units throughout hyperparameter tuning, as suggested by the grid search technique, to provide additional coverage to find the optimal solution.

Sequence length tuning rule: The sequence lengths are also adjusted once performance trends are observed. Consideration of the data set frequency and data ingest method is important when applying this tuning rule. For instance, this data set's primary recording rates are 16, 8, and 4 Hz, therefore recommended sequence lengths are equal to the frequencies found in the data. Otherwise, stale data created by the pad method for the 8 and 4 Hz signals can reduce overall performance of the model other sequence lengths are used.

D. Methodology verification

This methodology's selection accuracy is verified by using 20 random flights each from two other randomly chosen aircraft tail numbers within this data set. The verification executes the bootstrap loop in Fig. 2 using the random data from the other aircraft and three pre-trained models of varying performance from the selection process. The optimally selected pre-trained model and two other lower performing models are compared for relative performance based on average MSE. This verifies that the initially trained and selected model continues to outperform non-optimal pre-trained models on other aircraft with the same sensors.

V. Results

Executing this methodology on the NASA data set produced 248 unique aircraft altitude sensor fusion models before selecting an optimal solution. It started from an initial DL configuration matrix with 72 DL-approach and hyperparameter configurations then executed the methodology discussed in Section IV. The initial matrix was composed of the hyperparameters shown in Table III and all of the DL approaches listed in Table II. Table III also lists the final selected model's hyperparameters. Model training and testing of the DL configuration matrix for each iteration was run in parallel. Section V-A summarizes the results from all refinement iterations and the application of the rules. Section V-B presents the convergence of each refinement iteration's results. Lastly, section V-C presents the results from the methodology verification.

TABLE III
LSTM HYPERPARAMETERS

Config	SeqLen	LR	Epochs	Units
Initial Matrix				
1	8	0.01	1,000	10
2	8	0.05	1,000	20
3	16	0.01	1,000	10
4	16	0.03	1,000	20
5	16	0.05	1,000	20
6	16	0.05	1,000	25
Optimal Model (Iteration 4)				
31	16	0.0007	3,500	20

A. Refinement iterations

As mentioned previously, the first step of this methodology is to train and measure the performance of the DL configuration matrix and then use the average bootstrap loop MSE as inputs into the second iteration for pruning and tuning.

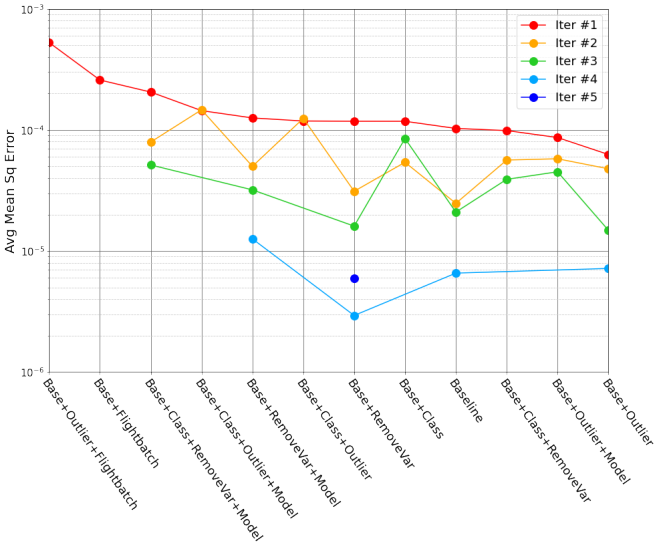

Fig. 3. Minimum MSE Across Refinement Iterations for each DL Approach

As expected, the average MSE generated from the first iteration did not meet the convergence criterion, therefore the methodology proceeded onto iteration two. In iteration two the *Prune_N_Tune* procedure pruned approaches 11 and 12 for being the worst two performing approaches. This is indicated in the $Pruned_i$ column of Table II and can be seen in Fig. 3 by the missing data points between the iteration 1 and 2 lines.

The best and worst performing hyperparameter configurations, one and five in Table III, respectively, were used to calculate the second iteration's learning rate and Epochs using equations (2), (3), and (4). An initial damping factor of 4 was chosen and is divided by 2 each successive iteration. Model convergence data indicated the number of initial epochs were larger than needed for convergence, therefore $Epochs_{i_{best}}$ was reduced before the calculations. No trends were identified within the hidden units, therefore additional values were added to expand exploration of the hyperparameters. No significant performance difference between sequence lengths were identified, therefore two additional sequence length configurations equal to 8 were added to expand coverage as well.

Iterations two through four proceeded in much the same way as iteration one. The worst two DL approaches were pruned and the hyperparameters tuned according to the associated equations. However, iteration four pruned an additional two DL approaches based on a trend of consistently low performance across iterations one through three. Also, iteration five proceeded with the single best performing approach. Fig. 3 displays the progression of each iteration's best tuned approaches as the GD-MAGS methodology is performed on the data. This figure also indicates performance differences for DL approach as the hyperparameters are tuned. This is observed in the inconsistent drop of MSE across DL approaches per iteration.

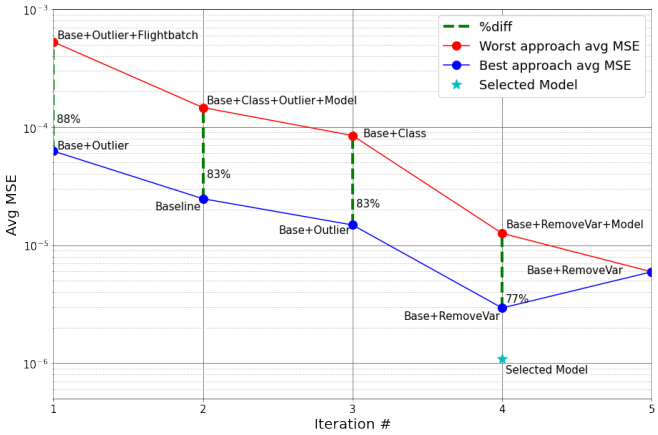

Fig. 4. Best and Worst DL approaches across iterations

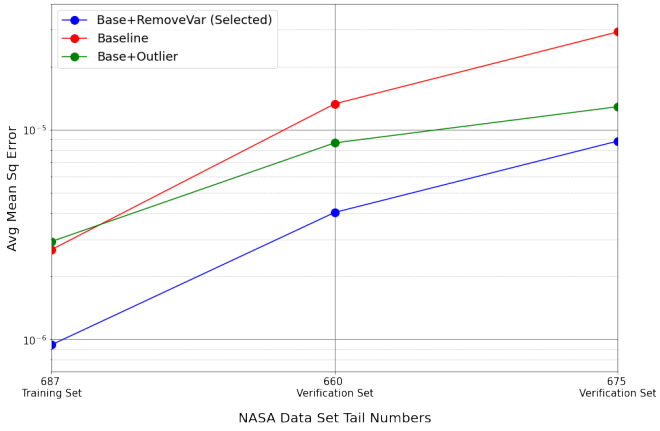

Fig. 5. Relative Performance Verification with Two Other Aircraft Data Sets

The results from iteration five returned a minimum MSE that was greater than iteration four's minimum MSE as seen in Fig. 4 by the rise in MSE at iteration five in the lower blue line. From this observation, it was determined that the minimum had been found in iteration four and the procedural loop was terminated. Ideally, additional iterations centered around iteration four's data would further refine the optimal configuration, however finding the location of a minimum was deemed sufficient for this study. Therefore, the *Base+RemoveVar* approach and hyperparameter configuration 31 from Table III was selected as the optimal model.

B. Convergence results

This section presents the selection methodology's convergence across all 12 DL approaches and the hyperparameter space. Fig. 4 shows convergence of the best and worst DL approaches and hyperparameters as the blue and red lines draw nearer to each other. The dotted green lines and percentage labels quantify the convergence between best and worst approaches at each iteration by listing the percent difference between them. Fig. 4 also shows the optimal bootstrap average at iteration four and the the MSE of the selected model from the best of the three bootstrap loop models. This figure also labels the best and worst performing DL approaches across the five refinement iterations. The final selected configuration in iteration four has a bootstrap loop average MSE of 2.93e-06 while the selected model has an MSE of 1.09e-06. When comparing error of altitude output from this selected model to the seven test files used in iteration four the maximum in-flight error was 155.4 ft as the aircraft is transitioning from level flight into a descent. Larger errors are present for this comparison but they occurred in transitions between disparate flight files and aircraft post-landing maneuvers which are artifacts of the batch process and classification, respectively. These errors were not considered as in-flight errors, but were included in the standard deviation. The standard deviation of the errors across all seven flights is 29.7 ft.

C. Verification results

The optimally selected model from the *Base+RemoveVar* approach and the two next highest performing models from iteration 4, *Baseline* and *Base+Outlier*, were selected as the models to use for model selection verification. These models were trained on tail number 687 and the verification testing was performed on tail numbers 660 and 675 which were all chosen randomly. These three pre-trained models ingested data from the training set and two other aircraft tail numbers as verification data sets. A bootstrap loop average for each model and data set was calculated and plotted in Fig. 5. This figure shows that the average bootstrap MSE for the optimally selected model continues to outperform the other two models on two other aircraft data sets.

VI. Discussion

The focus of this paper is the GD-MAGS optimal selection methodology for DL-based sensor fusion solutions across a range of DL approaches. The novel approach here focused on the system of combining several different data techniques, Neural Network algorithms and hyperparameter tuning together, rather than simply investigating a new method of Deep Learning algorithms alone. This methodology could have continued to iterate using the MSE gradient after iteration five and further refined the hyperparameters, but we decided this minimum was sufficient for demonstration purposes. The value of this methodology has been shown by its successful identification of the best DL approach and hyperparameter configuration. Part of this success can be shown by the removal of 6 DL approaches known to perform poorly. In one example of this, two under performing approaches were eliminated in the first iteration. The second worst approach, *Base+Outlier+Flightbatch*, only differed from the best approach, *Base+Outlier* by their respective batch sizes of 125,000 to 50,000 and had an 88% difference in average MSE as shown in Fig. 4. This matches observations made by Keskar et al. [17] that found when using a larger batch size, there is a degradation in the quality of the model, as measured by its ability to generalize.

In the second example, the methodology pruned another four approaches when it correctly identified the lower performing *Model* technique approaches. A key assumption used in this technique was discovered late in this study to be wrong. We incorrectly assumed, initially, that the RALT measurement indicated an aircraft terrain following feature that kept it roughly 5,500 ft above the earth, instead, this was a limitation of the range of the radar altimeter sensor. Therefore, the data provided by this technique added meaningless and possibly misleading data to the LSTM algorithm which proved to affect the model's accuracy. The selection methodology correctly pruned the DL approaches with this technique and demonstrated value in removing an under-performing approach.

Two possible challenges of implementing this methodology is balancing the correct number of bootstrap loops and number of flight files to use. Increasing bootstrap loops or files slows down the selection process significantly. Too few bootstrap loops or files introduces the risk of stochastic results or poorly generalized models. We verified our choices of bootstrap loops and number of files used in this demonstration by analyzing the average MSE for 20 bootstrap loops and doubling the number of test files for the selected model. This showed that the average MSE with three bootstrap loops vs 20 only differed by 1.04e-07. It also showed that the average MSE using 14 testing files, instead of the 7 done in this paper, for 20 bootstrap loops only differed by 2.34e-08. These values are less than the difference between the best and second best model average MSE and indicates using 7 testing files for 3 bootstrap loops was sufficient for this data set. However, we recommend users perform similar studies on their data.

VII. Conclusions and Future Work

The GD-MAGS selection methodology successfully demonstrated it is capable of selecting a single DL approach out of many and while tuning hyperparameters to generate an optimal sensor fusion model for a system of multi-modal sensors. This methodology provides a reliable process that many engineers can apply to sensor data fusion problems with the knowledge it will converge on an optimal solution. Our methodology has also verified that its selected sensor fusion model generalizes well to other aircraft tail numbers with the same sensors. This verification provides a compelling justification that this optimization methodology can be applied across many data sets and the selected optimal model can be applied to characteristically similar sensor fusion problems. The generalization of this methodology also applies to other real-time sensor integration problems outside of the aircraft industry. Our demonstration and experience with this methodology has provided valuable insights into possible areas of advancement for sensor fusion.

It is currently unknown if refining each DL approach's hyperparameters individually, rather than all together, would provide a faster convergence to the optimal solution. Future work will investigate this and will also pursue a software framework to be used by other researchers. Additional work will apply this selection methodology to different types of sensors and other target variables to measure its generalization performance on more complex sensor fusion problems. This will include implementation of additional DL algorithms, ensemble networks and a variety of other data ingestion techniques from current sensor fusion literature. Future work is aimed at ingesting DL-based sensor fusion algorithm's information into a SysML system model of a resource limited sensor suite towards analysis of design and architecture optimizations and Pareto trade-offs.

Acknowledgment

This work was supported in part by funding from NSF under Award Number OAC 1931363, the US Air Force STEM+M program, the AFSC/309th Software Engineering Group, and NexusDE LLC.

References

[1] B. Khaleghi, A. Khamis, F. O. Karray, and S. N. Razavi, "Multisensor data fusion: A review of the state-of-the-art," *Information Fusion*, vol. 14, no. 1, pp. 28–44, 2013.

[2] S. Makridakis, E. Spiliotis, and V. Assimakopoulos, "The M5 accuracy competition: Results, findings and conclusions," *Int J Forecast*, no. October, pp. 1–44, 2020.

[3] I. M. Pires, N. M. Garcia, N. Pombo, and F. Flórez-Revuelta, "From data acquisition to data fusion: A comprehensive review and a roadmap for the identification of activities of daily living using mobile devices," *Sensors (Switzerland)*, vol. 16, no. 2, 2016.

[4] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016.

[5] NASA, "DASH Link," 2018.

[6] B. N. Oreshkin, D. Carpov, N. Chapados, and Y. Bengio, "N-BEATS: Neural basis expansion analysis for interpretable time series forecasting," in *International Conference on Learning Representations*, 2020.

[7] D. Salinas, V. Flunkert, J. Gasthaus, and T. Januschowski, "DeepAR: Probabilistic forecasting with autoregressive recurrent networks," *International Journal of Forecasting*, vol. 36, no. 3, pp. 1181–1191, 2020.

[8] B. Lim, S. Arık, N. Loeff, and T. Pfister, "Temporal Fusion Transformers for interpretable multi-horizon time series forecasting," *International Journal of Forecasting*, vol. 37, no. 4, pp. 1748–1764, 2021.

[9] H. J. P. Weerts, A. C. Mueller, and J. Vanschoren, "Importance of Tuning Hyperparameters of Machine Learning Algorithms," *arXiv*, 2020.

[10] J. Bergstra, D. Yamins, and D. D. Cox, "Making a Science of Model Search: Hyperparameter Optimization in Hundreds of Dimensions for Vision Architectures," *Proceedings of the 30th International Conference on Machine Learning*, vol. 28, 2013.

[11] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization," *Journal of Machine Learning Research*, vol. 13, pp. 281–305, 2012.

[12] S. M. Howard and M. S. Lewicki, *Deep Learning for Sensor Fusion*. PhD thesis, Case Western Reserve University, 2017.

[13] S. Yao, S. Hu, Y. Zhao, A. Zhang, and T. Abdelzaher, "DeepSense: A unified deep learning framework for time-series mobile sensing data processing," *26th International World Wide Web Conference, WWW 2017*, pp. 351–360, 2017.

[14] M. Mitchell, S. Wu, A. Zaldivar, P. Barnes, L. Vasserman, B. Hutchinson, E. Spitzer, I. D. Raji, and T. Gebru, "Model cards for model reporting," *FAT* 2019 - Proceedings of the 2019 Conference on Fairness, Accountability, and Transparency*, no. Figure 2, pp. 220–229, 2019.

[15] S. Liu, H. Liu, V. John, Z. Liu, and E. Blasch, "Enhanced situation awareness through CNN-based deep multimodal image fusion," *Optical Engineering*, vol. 59, no. 05, p. 1, 2020.

[16] D. Danielson, J.J., Gesch, "Global Multi-resolution Terrain Elevation Data 2010 (GMTED2010)," *U.S. Geological Survey Open-File Report 2011-1073*, vol. 2010, p. 26, 2011.

[17] N. S. Keskar, J. Nocedal, P. T. P. Tang, D. Mudigere, and M. Smelyanskiy, "On large-batch training for deep learning: Generalization gap and sharp minima," *5th International Conference on Learning Representations, ICLR 2017 - Conference Track Proceedings*, pp. 1–16, 2017.