Exploring Compositional Neural Networks for Real-Time Systems

Sobhan Chatterjee[*] [†], Nathan Allen[*], Nitish Patel[*] and Partha Roop[*]
[*]*Department of Electrical, Computer and Software Engineering, Faculty of Engineering*
University of Auckland, Auckland, New Zealand
[†]*Email: schb534@aucklanduni.ac.nz*

***Abstract*—Real-time CPSs using Artificial Neural Networks (ANNs) are traditionally developed as monolithic black-boxes. This results in designs that are often difficult to formally verify against safety specifications and implement on hardware for formal timing analysis. Consequently, their implementation as a composition of smaller ANNs has received recent interest. These are easier to implement, parallelise and validate. Despite this, the question of how to produce hardware-implementable compositional designs from existing monolithic ones remains largely unanswered.**

This work develops a novel procedure to replace large ANN monolithic designs with smaller compositional designs and implement them on a Field Programmable Gate Array (FPGA) for timing analysis using synchronous compositional semantics. To illustrate our approach, we develop regression and classification ANN designs for multiple real-life datasets. Using various design and model architecture variations, we show that using a compositional design instead of a monolithic design can achieve an 85% reduction in WCET, around a 53% reduction in hardware resources and around a 40% reduction in computations and neuron connections for a minor reduction in performance.

1. Introduction

Cyber-Physical Systems (CPSs) are inherently compositional. Generally composed of controllers distributed across physical space and hardware resources, CPSs are used to design real-time systems with strict timing and functional requirements [1]. Traditionally, CPS design employs model-driven techniques, where one designs a complex system as a combination of several interacting components (controllers and controlled systems). Such an approach facilitates the design of simpler components, which are easier to validate and integrate with the overall system [2].

However, as systems become more complex with time, model-driven design based on physical process modelling is becoming increasingly difficult. Consequently, data-driven techniques for system design, typically Artificial Neural Networks (ANNs), have been gaining prominence as the alternative form of CPS design due to their ability to model complex input-output relationships without extensive information about the system dynamics [3]. Nevertheless, ANNs are often prone to errors [4] and their use in real-time systems with strict safety (safety-critical) and response-time (time-critical) specifications often mandate formal guarantees on their functional [5] and timing correctness [6].

While several techniques have been proposed to handle formal verification [7] of ANNs, which is an NP-Complete problem [8], the use of such methods is usually limited to either small networks or require complex model abstraction techniques for larger networks. This limitation primarily results from ANNs being designed and implemented as monolithic black boxes [9]. Monolithic designs tend to be big, slow to execute, resource-intensive, and seldom lend themselves to parallel or incremental designs [1]. Moreover, the size of monolithic designs also restricts their implementation on hardware to smaller networks with just tens of neurons, as evident from [10] and [11]. Consequently, Worst Case Execution Time (WCET) analysis, defined as the maximum time a program takes to execute on a specific hardware platform and is one of the ways of ensuring the timing correctness of real-time systems, is challenging for monolithic ANNs.

A compositional approach, which advocates using smaller but functionally equivalent systems to the ANN-based CPS design, is ideal for addressing the problems with large designs. Indeed [1] presents the first compelling case for compositionality in systems in general. Although recent work on data-driven CPS design has tried to tackle the notion of compositionality in many ways, studies that comprehensively examine compositional ANNs for their functional verification and timing correctness are lacking.

While some studies like [12], [13], [14] use compositional designs to facilitate verification, they focus on systems in which an ANN is a component of the compositional design and not the focus of the compositional design itself. Moreover, a few works, such as [15] and [9], have tried to address the WCET analysis of ANNs using compositional designs on hardware. However, their methods do not focus on using compositional models to reduce the WCET of monolithic models. On the other hand, orthogonal studies, such as [16], [17], that use model reduction techniques instead of compositionality cannot guarantee a reduced network, making their hardware implementation unreliable. Overall, there is a noted lack of research that systematically, quantitatively, and objectively presents how a large monolithic ANN model can be replaced with several smaller networks (a compositional design) to ease their implementation on hardware for timing verification using WCET.

Hence, in this work, we design multiple examples to comprehensively quantify the benefits of a novel compositional design for hardware implementation over a monolithic design. We use several real-life datasets for the examples, including a depression dataset and a sizeable transportation dataset - NGSIM (Next Generation SIMulation) - to develop the models that can help cars perform Discretionary Lane Changing (DLC).

Through these examples, we make the following contributions. 1) We present a detailed methodology of how a Multilayer-Perceptron (MLP) based feed-forward monolithic design can be replaced with smaller models that combine to form a compositional design that reduces the hardware footprint of the model at minimal to no performance loss. The method presented is not specific to the datasets and type of ANNs considered in this work and can be used with most feed-forward ANNs with minor to no changes. 2) We describe how the compositional design can be implemented on hardware for proper timing analysis. 3) We quantitatively examine the monolithic and compositional designs using software models to test their performance, the hardware models to compare the WCET, hardware logic requirements and their verification captured by comparing neuron connections and computations required for inference. We demonstrate that, for the DLC dataset, using a compositional rather than a monolithic design can achieve significant, up to 85% reduction in WCET, around a 53% reduction in hardware resources and around a 40% reduction in computations and neuron connections for a minor reduction in performance.

The paper is structured as follows. Firstly, in Section 2, we introduce a brief background on the example datasets considered. Later, in Section 3, the compositional premise used in this work is introduced. Subsequently, in Section 4, we provide the details on the development of the designs. Section 5 describes the hardware implementation and the semantics used. Next, we present the comparison reports and discussions in Section 6 to show how the compositional models compare to the monolithic models. Finally, Section 7 contains concluding remarks and the study's future directions.

2. Background

2.1. Artifical Neural Networks

Artificial Neural Networks (ANNs) are, as the name suggests, networks of artificial neurons that attempt to model the behaviour of biological neurons using linear and nonlinear mathematical functions. The nonlinear functions are called activations. The ANN training is data-driven, and once trained, it can make inferences on new but statistically similar input data [3]. They are often designed in a layered structure, with an input layer, a few hidden layers, and an output layer.

2.2. Depression Studies

Depression is a disorder involving loss of pleasure or interest in activities for long periods and is associated with sustained mood deterioration [18]. The World Health Organisation (WHO) 2023 estimates that 5% of adults (approximately 300 million) worldwide experience depression. Recent depression research has focused on using wearable technology to assess patient mood as it is unobtrusive, real-time and often passive and allows assessments to occur in the person's usual environment [19]. Research has now grown in the direction of personalised, predictive models for depression to account for the substantial inter-individual variability in depressive symptoms, and the results are promising [20].

2.3. Discretionary Lane Changing

A discretionary lane change (DLC) occurs when a driver follows another vehicle at speeds slower than their desired speed and wants to increase their speed by moving to an adjacent lane [21]. The study by Gipss [22] is one of the first to systematically describe the structure of the driver's overall lane change (LC) decision-making process. More recently, there has been a noticeable shift towards developing data-driven, predominantly ANN-based models for LC. The goal is to make models learn the complex mathematical functions that describe a safe lane-changing process by observing data on parameters that serve as inputs to the model.

Generally, the input parameters that govern the learning process include the speed of the subject and surrounding vehicles and the gaps or spacing between the subject vehicle and surrounding vehicles. Figure 1 shows a typical arrangement of cars before a lane change and the parameters on which the lane change may depend. Similar to studies such as [23] and [24], in this work, we consider the vital decision-making module for DLC as our example to demonstrate the development and use of the compositional design.

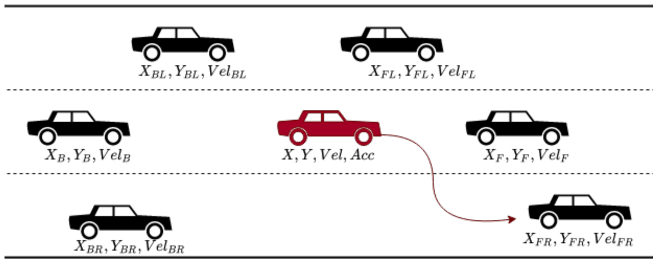
X_{BL}, Y_{BL}, Vel_{BL}
X_{FL}, Y_{FL}, Vel_{FL}
X_B, Y_B, Vel_B
X, Y, Vel, Acc
X_F, Y_F, Vel_F
X_{BR}, Y_{BR}, Vel_{BR}
X_{FR}, Y_{FR}, Vel_{FR}

Figure 1: Defining the features in DLC. B, F, L, and R stand for Back, Front, Left and Right. These terms specify the position of the car. We take the values of these features relative to the ego/subject vehicle.

2.4. Datasets

We test our compositional framework on various real-life datasets, as presented next.

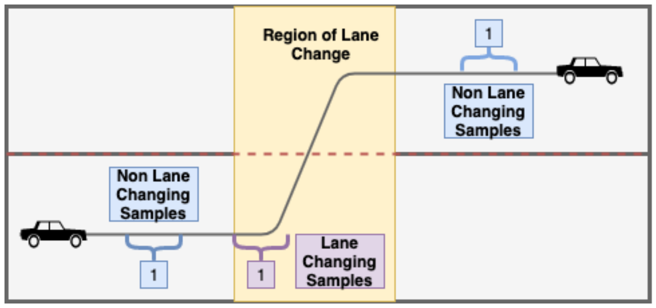

Figure 2: Extracting samples from NGSIM dataset.

2.4.1. Small Datasets. We use three real-life datasets to build example ANNs demonstrating the compositional approach. We choose the famous Iris flower classification dataset [25], a Diabetes dataset [26], and a Wine classification dataset [27] for this task and build ANN classification designs for them. The Iris dataset has 150 samples (and four features/inputs), the Diabetes dataset has 442 samples (and ten features), and the Wine dataset has 178 (and 13 features) samples. Although the Diabetes dataset is a regression problem, we transform it into a classification problem by clustering it into three groups to form a new target/output feature. Overall, all three examples have three output classes/values.

2.4.2. Depression Dataset. We use a dataset previously published in [20] to build regression-based personalised (one model per subject) mood score prediction models. This dataset was collected during a one-month study of 14 mild-moderately depressed adult human subjects (with a mean age of 21.6 ± 2.8 years and ten females) before the onset of the COVID-19 pandemic. Although the dataset contains data for 14 individuals, we use only seven to keep the results concise.

The dataset includes data collected from smartwatches, smartphones and clinical EEG-based neuro-cognitive evaluations. The collected mood score values - lie between 1 and 7, with 1 for feeling not depressed and 7 for feeling severely depressed. We preprocess the raw dataset containing 48 features (or predictors) using an imputation method described in [28] and obtain 43 input features (i.e. inputs to a model), one output feature (mood scores) and one feature to preserve timing information. The amount of data per human subject lies between 34 and 120. More information on the features can be found in [28].

2.4.3. Next Generation SIMulation (NGSIM). We use the well-known NGSIM database [29] for the DLC example. The database contains, among other datasets, vehicle trajectory data gathered at a segment of I-80 Freeway in Emeryville, California and a segment of US Highway 101 in Los Angeles, California. The dataset is substantive, with over six million data points collected for over 10000 vehicles, but is rather noisy. Hence, we filter the dataset using the method proposed in [30]. Samples are then extracted from the filtered dataset - a process justified in [31].

For this extraction, we first find the six vehicles (cars or not) surrounding a lane changing (*ego*) car and extract 1-second regions (10 data points) from around the time an ego begins to lane change and also 1-second regions from before and after the lane change to include samples that correspond to feasible or infeasible conditions for a lane change respectively (Figure 2). The extracted dataset contains 16,676 sample points from 559 lane changing cars, each sample containing 22 input features and three output classes. There are 11,006 No LC (lane keep) samples, 4380 Left LC samples and 1290 Right LC samples.

3. The Compositional Premise

The compositional premise used in this research relies on the divisibility of existing monolithic feedforward ANN-based models to reduce hardware implementation costs at minimal to no performance loss. We focus on feedforward ANNs as they are the most widely used ANN architecture.

Although our approach allows for arbitrary combinations of smaller feedforward compositional models (especially Multilayer Perceptron (MLP)) to replace a large monolithic model, which is too big to implement on hardware or verify, the work we present in this paper limits itself to a two-stage architecture of compositional design with MLP models for brevity. Arbitrary combinations of models are possible as long as the input-output compatibility of the stages is maintained.

Moreover, the approach differs in design between classification and regression problems. Nevertheless, the underlying two-stage architecture remains the same. The following sections describe the compositional premise for classification and regression problems separately.

3.1. Compositional Design for Classification

We rely on the divisibility of an existing monolithic feedforward ANN-based MLP classification model on the output class or label that the model predicts. Specifically, we decompose or divide a monolithic design with C output classes into $C-1$ compositional models, where each model corresponds to a particular class. These models are trained such that a model corresponding to class C will output a High (valued 1) when the input to the model has the same class label; otherwise, the model will output a Low (valued 0). We use $C-1$ as opposed to C models as we believe (and later show) that this would lead to a better trade-off between a model's predictive performance and hardware implementation cost.

The compositional models exist in a single stage and execute in parallel. The outputs from these models are then passed onto a merge block that interprets these outputs and generates a single output value corresponding to any of the C class labels. Although we can use any mathematical function (including an ANN model) to combine the outputs from the compositional models, we propose a simple conditional merge block to simplify hardware implementation. As per this conditional merge block, when only one of the $C-1$

compositional models produces a High output (valued 1), the merge module passes on the label for that model's class. However, when more than one model passes a High output, or none of them passes a High output, the label of the remaining C^{th} class is passed as the output. This method enables the compositional classification of datasets with C classes with just $C-1$ models, as shown in Figure 3.

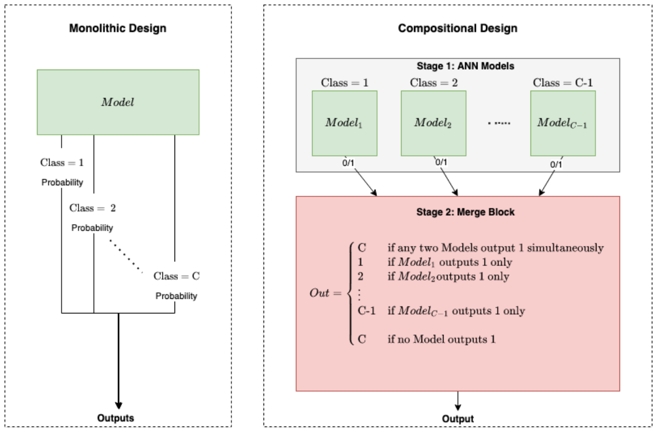
Monolithic Design
Model
Class = 1
Probability
Class = 2
Probability
Class = C
Probability
Outputs
Compositional Design
Stage 1: ANN Models
Class = 1
Class = 2
Class = C-1
$Model_1$
$Model_2$
$Model_{C-1}$
0/1
0/1
0/1
Stage 2: Merge Block
$Out = \begin{cases} C & \text{if any two Models output 1 simultaneously} \\ 1 & \text{if } Model_1 \text{ outputs 1 only} \\ 2 & \text{if } Model_2 \text{ outputs 1 only} \\ \vdots & \\ C\text{-}1 & \text{if } Model_{C-1} \text{ outputs 1 only} \\ C & \text{if no Model outputs 1} \end{cases}$
Output

Figure 3: Single-stage monolithic design and a 2-stage compositional design. We assume that we build models for the classes labelled $1, 2, \cdots, C-1$, and the C^{th} class is deduced as shown.

We build the compositional models separately from the monolithic models and then combine the trained models using the merge block. The division of the monolithic model into $C-1$ classes can be based either on domain (data) knowledge or using qualitative and quantitative methods. For a more quantitative approach, we recommend using the $confusion-matrix$ we obtain from the monolithic design evaluation and use the diagonal of the $confusion-matrix$ computed. Then, the diagonal of the matrix can be sorted to find the top $C-1$ easily classifiable classes and build models for them. A higher diagonal value implies higher ease in classifying the corresponding class. Building models for the $C-1$ easily classifiable classes should ensure that the remaining C^{th} class is properly deduced by reducing the number of false positives from the trained models.

Similarly, the division of neurons among the $C-1$ models can be based on intuition/domain knowledge or a quantitative approach. We propose beginning with dividing the neurons equally between the chosen models and then deciding whether to optimise the neuron division among the models based on the performance of the compositional design vis-à-vis the monolithic design. A designer may choose a threshold for performance loss observed (if any) when migrating to a compositional design before opting for the optimisation route. This ensures that the compositional design is not over-engineered and that the performance loss is minimal.

We use the optimisation approach if the equally divided compositional design has a $Precision$ loss of more than one percent. We optimise the compositional design $Precision$ using Bayesian optimisation with Gaussian processes [32] while ensuring that the total neurons in the compositional models remain the same. The optimisation can be stopped after a fixed number of trials or after the loss has attained a specific value. Also, any other suitable optimisation method, such as evolutionary algorithms, can be used in place of the one proposed, but the goal of optimising the model performance remains the same.

In the general case, all the compositional models will share the same set of input features, i.e., those used by the monolithic design. However, if data domain knowledge can be used to remove a few redundant features in a few compositional models, it should be used to reduce the number of neurons in the compositional models, which can be beneficial for hardware implementation.

3.2. Compositional Design for Regression

The compositional design for regression problems is similar to the classification problem, with a few differences. Since there are no classes in a regression problem, we do not divide the monolithic MLP design based on the values of the output feature. Here, we exploit the divisibility of the input feature set into individual compositional models. The division must be based on either data domain knowledge or a quantitative approach. We propose using data domain knowledge first, followed by quantitative methods, if the former is not feasible or does not achieve the same level of performance as the monolithic design.

Similar to the classification problem, the compositional design for the regression problem is also a two-staged design, with the first stage containing the compositional models (running in parallel) and the second merge stage combining the several model outputs into one. The individual compositional models output a single continuous value. These are then combined using a mathematical function. We propose a weighted combination of the outputs from the compositional models as shown in Equation 1 for a simpler hardware implementation.

$$output\_value = \sum_{i=1}^{n} w_i \times output_i \tag{1}$$

As we are dividing based on the features, the models in the compositional design will have a different number of features than the monolithic model, but the total number of features entering both designs will remain the same. The divided feature sets can be mutually exclusive or have repeated features. However, we advise using a set of mutually exclusive features to prevent feature redundancy, if that is desired, in the compositional models when we combine the models in the second step of the compositional design.

4. ANN Model Development

We follow the following steps for MLP-based ANN design development of the datasets described. First, we preprocess the data with standard normalisation/scaling if it

TABLE 1: **Search values (or limits) for parameters during hyperparameters optimisation of the Depression Dataset**. (We use N.A (Not Applicable) for cells with no value. The values are the ranges and values used for hyperparameter search.)

Hyperparameter	Monolithic			Compositional		
	Lower Limit	Upper Limit	Values	Lower Limit	Upper Limit	Values
Number of layers	2	4	N.A	2	4	N.A
Neurons in input layer	50	90	N.A	10	15	N.A
Activation	N.A	N.A	tanh, relu, elu, linear	N.A	N.A	tanh, relu, elu, linear
Batch size	N.A	N.A	4, 6, 8	N.A	N.A	4, 6, 8

is not normalised. Second, we train the monolithic design model for different ANN architectures, as we do not have existing monolithic models for any of the example datasets considered. Third, we build the compositional design architecture. When building the compositional design models, we can choose to have the same number of total neurons or different numbers in the two designs. While the former ensures a fairer performance comparison, we experiment with both approaches to comprehensively evaluate the two-stage compositional design. Finally, we train the compositional design with corresponding datasets using a 5-fold cross-validation (CV) method [3].

Additionally, we adopt best-model saving and learning rate reduction practices to alleviate the undesirable effect of stochasticity in training. Furthermore, the compositional and monolithic model training is performed in Python using the well-known TensorFlow-Keras Library [33] on a 64-bit AMD Threadripper 3990 with 64 cores and 256 GB RAM.

4.1. Small Dataset Models

We use a four-layer classification architecture for all monolithic and compositional designs for all three datasets. We ensure that the monolithic and compositional designs have the same number of total neurons. The Wine and Diabetes dataset monolithic designs have 24, 12, 6, and 3 neurons, while the iris dataset monolithic model has 20, 10, 4 and 3 neurons in the four layers from input to output. Next, we find the classes to build the compositional models using the quantitative method mentioned in Section 3.

We build compositional models for classes 0 and 1 for the Wine and Diabetes classes and classes 0 and 2 for the Iris dataset. We do not optimise the neuron division; instead, we divide the neurons equally between the two compositional models, as there was no performance loss. Further, we use $tanh$ for all layers except for the last layer, which uses $softmax$ for the monolithic model and $sigmoid$ for the compositional models. Also, both designs are trained for 200 epochs and a batch size of 8 for the Wine and Diabetes datasets and 4 for the Iris dataset.

4.2. Depression Dataset Models

We build personalised regression models for the Depression dataset, which contains data from seven human subjects. As it is usually challenging to build models on data collected from human biophysical signals due to their complexity, models often need to be optimised for their performance. Instead of manually choosing and tweaking a few parameters to obtain better performance, we chose Evolutionary Algorithm (EA) based algorithms to optimise the number of hidden layers in the model, the number of neurons in the input layer, the activation of the hidden layers, and the training batch size for both monolithic and compositional models. The designs are optimised for 50 iterations using Mean Absolute Error (MAE) as the performance metric.

Once we have optimised the monolithic models, we divide the input feature into six mutually exclusive sets. We divide the features based on whether they refer to food, activity, sleep, physiological, psychological or neurocognitive feature sets and build compositional models for the sets. A compositional design contains these six models, which are trained and optimised separately and then combined using the merge block, the weights of which are learned using a gradient descent-based algorithm. The search parameters for the optimisation can be found in Table 1. Regardless of the optimisation procedure, as we are making models for a regression problem, the final layer in the models for both designs always uses the $linear$ activation function.

Furthermore, we do not ensure that the compositional and the monolithic designs have the same total neurons. However, we ensure that the neuron ranges for the compositional models are nearly a sixth of the monolithic model. This difference helps ascertain how well the compositional approach works when the total number of neurons in the compositional design differs from the monolithic design.

TABLE 2: **Best Neuron Divisions (DLC Dataset)**. (TN refers to the total number of neurons in the model. Left and Right refers to the compositional models for the Left and Right lane change classes. The values are the fraction of neurons in each model after optimisation.)

	1 Layer			3 Layers		
Models ↓ / TN →	118	157	195	118	157	195
Left	0.69	0.71	0.74	0.62	0.54	0.51
Right	0.31	0.29	0.26	0.38	0.46	0.49
	6 Layers			10 Layers		
Models ↓ / TN →	118	157	195	115	157	192
Left	0.58	0.60	0.64	0.70	0.58	0.58
Right	0.42	0.40	0.36	0.30	0.42	0.42

4.3. DLC Dataset Models

We build different monolithic and compositional classification designs for the DLC dataset, each with a different architecture. The Left and Right lane change (LC) classes are chosen for the compositional design based on observed

DLC behaviour, and the No lane change class is left out. Moreover, we experiment with three values of total neurons, i.e. {118, 157, 195}, and different numbers of layers, i.e. {1, 3, 6, 10}, in the monolithic and compositional designs.

Furthermore, we optimise the neuron division as the equal division of neurons between the models did not yield designs with good compositional performance. We optimise the designs using Bayesian optimisation based on the $Precision$ of the compositional designs. We end the optimisation after 50 trials and use 20 initial points with LHS. Table 2 contains the near-optimal division (fraction of neurons in each model) of neurons in compositional design obtained after optimisation as mentioned in Section 3. We also experiment with a design with three models (C models) and neurons equally divided neurons among them for a more thorough comparison.

Moreover, while we have 22 features for the monolithic examples, we remove the features corresponding to cars on the right lane for the Left LC Model, and for the Right LC Model, we do the vice-versa. This feature reduction is justified as the vehicles travelling in the right lane have a limited impact on a driver's decision to change lanes to the left and vice-versa. This reduction results in each compositional model having just 16 features. We should, however, note that the compositional design, on the whole, has the same number of inputs as the monolithic model.

5. Hardware Implementation

We implement the monolithic and compositional designs on hardware to analyse their timing performance through WCET analysis. Therefore, we develop a compiler to compile the ANN designs (TensorFlow-Keras [33] and Pytorch [34] ANN designs) to Very high-speed integrated circuit Hardware Description Language (VHDL) for implementation on a Field Programmable Gate Array (FPGA) through a process adapted from [15]. In contrast to the work in [15], we develop the compiler in Python. This allows the compiler to be *imported* (as a module) into existing Python-based ANN projects for easy compilation of ANN models. The compiler allows for arbitrary combinations of feedforward MLP-based ANN models developed using TensorFlow and Pytorch, including the 2-stage design we use.

5.1. Compilation of ANN design to VHDL

Algorithm 1 shows the high-level pseudo-code for the compiler. It converts an ANN design composed of single or multiple ANN models to a VHDL design with multiple files using an intermediate JSON design representation to represent the hierarchical inter-model (between models) and intra-model (with a model, i.e. between model layers) dependencies and connections.

The $nn2VHDL$ function is the compiler's entry point. It accepts models ($models$), their interconnections ($conn$), the number of pieces/parts to use for the piecewise linear activation function approximation ($aParts$), and the fixed-point representation bit size as inputs ($bSize$). It then passes these arguments to a parser that parses the design into a JSON representation ($jModel$). The activation approximation is implemented through Look-Up-Tables (LUTs). Following this, the representation is passed to a VHDL mapper ($writeVHDL$) that uses the model and layer connectivity information in the JSON representation to map the design to VHDL. The ANN model computations are combined into a single VHDL design file (using $genModPipelineMap$), while the activation functions are written into separate VHDL files (using $genActVHDL$) and are used as components in the VHDL design file. Also, $genActVHDL$ is only called once for each activation. A library VHDL file is also produced to perform fixed-point computations (using $genLibrary$).

Furthermore, the VHDL mapping (using $genModPipelineMap$) is based on a synchronous multicyclic and pipelined approach that enables the reuse of neuron instances mapped into hardware to reduce the

Algorithm 1 Compile ANN Designs to VHDL

```
1: function NN2VHDL($models$, $conn$, $aParts$, $bSize$)
2:     $jModel \leftarrow$ PARSE($models$, $conn$, $aParts$, $bSize$)
3:     $vhdlFiles \leftarrow$ WRITEVHDL($jModel$)
4:     return WRITEVHDLFILES($vhdlFiles$)
5: end function
6: function PARSE($models$, $conn$, $aParts$, $bSize$)
7:     $jModel \leftarrow \{\}$
8:     for $model, model\_name,$ in $models$ do
9:         $mA \leftarrow$ GETMODELARCHITECTURE($model$)
10:        $mW \leftarrow$ GETMODELWEIGHTS($model$)
11:        $mAc \leftarrow$ GETPIECEWISEACT($model$, $aParts$)
12:        $mLc \leftarrow$ GETLAYERCONNECTION($model$)
13:        $mJ \leftarrow$ {"architecture" : $mA$, "weights" :
               $mW$, "act_parts" : $mAc$, "layer_conn" : $mLc$}
14:        $jModel[model\_name] \leftarrow mJ$
15:    end for
16:    $jModel$["conn"] $\leftarrow conn$
17:    $jModel$["fp_size"] $\leftarrow bSize$
18:    return $jModel$
19: end function
20: function WRITEVHDL($jModel$)
21:    $modelVHDL \leftarrow []$
22:    $aFile \leftarrow []$
23:    for $conn$ in $jModel$["conn"] do
24:        for $model\_names$ in $conn$ do
25:            $mJ \leftarrow jModel[model\_names]$
26:            $mVHDL \leftarrow$ GENERATEVHDL($mJ$)
27:            append GENACTVHDL($jModel$["$mAc$"])
                 to $aFile$
28:        end for
29:        append $mVHDL$ to $modelVHDL$
30:    end for
31:    $mFile \leftarrow$ GENMODPIPELINEMAP($modelVHDL$)
32:    $lFile \leftarrow$ GENLIBRARY($jModel$["fp_size"])
33:    return $[mFile, aFile, lFile]$
34: end function
```

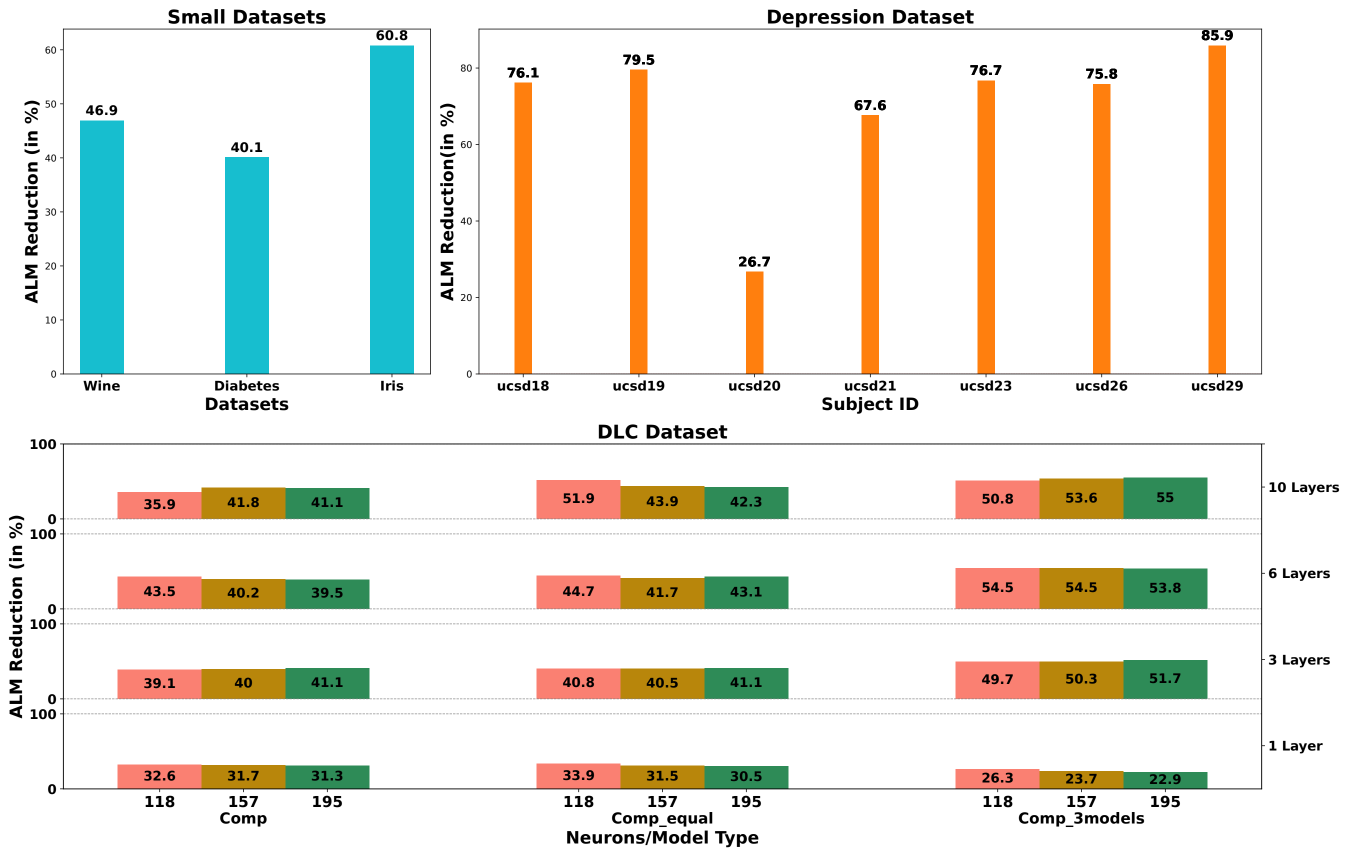

Figure 4: ALM reductions (in %) for all datasets. Each bar plot refers to each small dataset in the Small Datasets plot. The bars in the Depression Dataset plot correspond to the seven human subjects (with anonymised names) considered. The DLC dataset plot is shown for various total neuron numbers, layer counts, and model combinations. ”Comp” stands for the optimised compositional design, ”Comp-equal” stands for the compositional design with an equal number of neurons between the compositional models, and ”Comp-3models” stands for when the compositional design has three models with an equal number of neurons.

total hardware components needed for mapping. In this approach, a neuron is instantiated for each activation in hardware. A neuron reads a set of internal variables, performs computation, and then writes the updated output back to memory within one cycle. Subsequently, a counter is incremented, which causes the neuron to read the next set of variables from memory in the next clock cycle. Once all required computations have been performed, the execution is complete, signifying the end of the tick. This will cause variables to be mapped to their subsequent neurons in the next layer and allow execution for the next tick to begin. The number of cycles needed for a single execution is proportional to the maximum number of neurons of a particular activation.

The multicyclic approach enables us to fit networks larger than would usually be possible but at the cost of increased computation time. Since modern FPGAs have reasonably high clock frequencies, the increase in computation time should be minimal. The formal semantics for the multicylic approach can be found in [15].

6. Results

This section presents the results of the monolithic and compositional designs. We quantify the benefits and limitations of a compositional design against a monolithic design by comparing the predictive performance using $F1-score$ for classification models and MAE (Mean Absolute Error) for regression models, the hardware usage in terms of Adaptive Logic Modules (ALMs) and the Worst-Case Execution Time (WCET) for the hardware implementations, and the number of computations and connections in the design.

$F1-score$ is the harmonic mean of the $Recall$ and $Precision$, and produces a real number between 0 and 1. The MAE, as expected, is a real number over the entire real number domain. The performance of a classification (or regression) design is better if a design has a higher (or lower) $F1-score$ (or MAE) value. Equations 2, 3, and 4 define the $F1-score$, $Precision$, and $Recall$ metrics, respectively. Since we use a 5-fold CV approach, we average the performance metrics across the five folds before the $F1-score$ and MAE calculation.

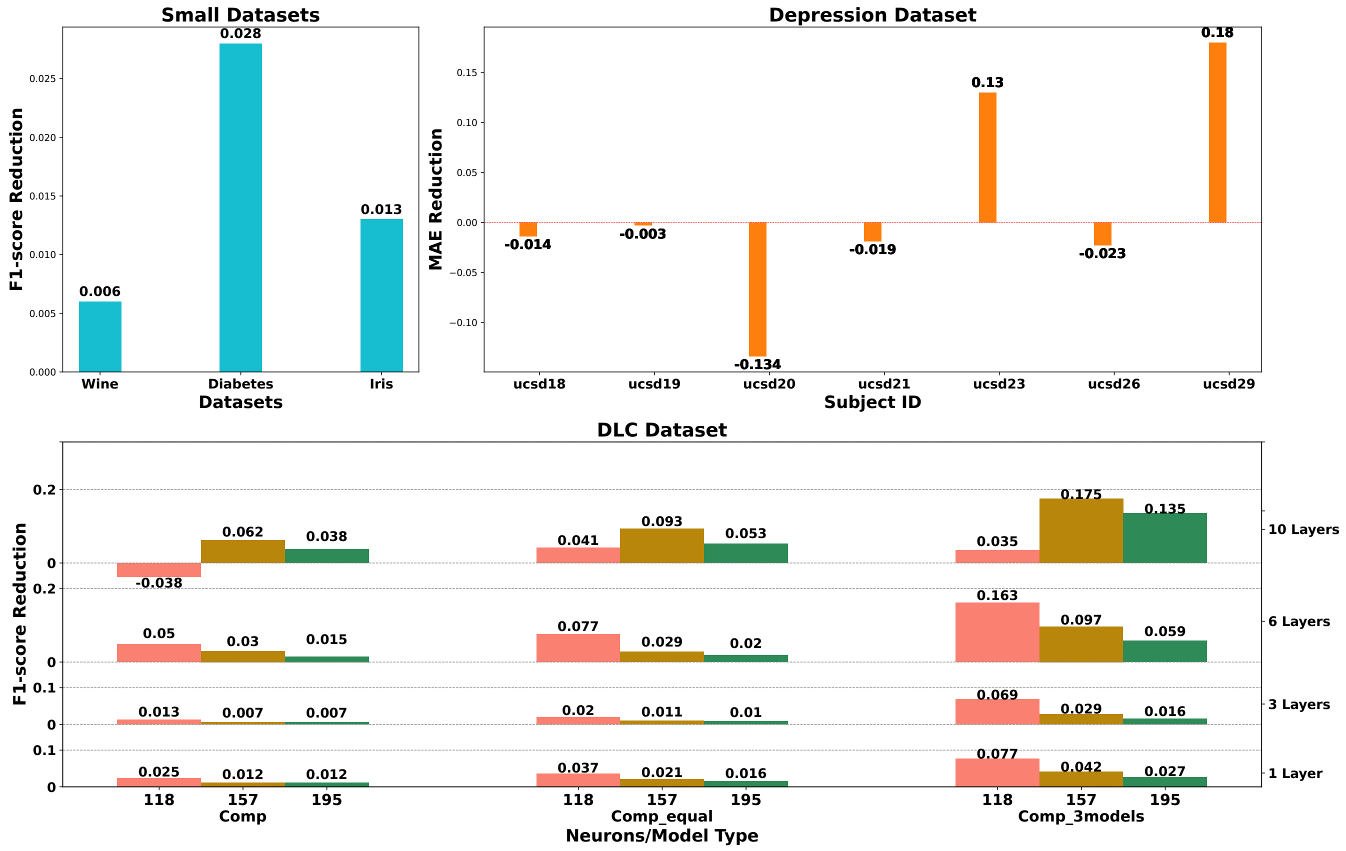

Figure 5: Predictive performance reduction (in absolute terms) for all datasets. Each bar plot refers to each small dataset in the Small Datasets plot. The bars in the Depression Dataset plot correspond to the seven human subjects (with anonymised names) considered. The DLC dataset plot is shown for various total neuron numbers, layer counts, and model combinations. "Comp" stands for the optimised compositional design, "Comp-equal" stands for the compositional design with an equal number of neurons between the compositional models, and "Comp-3models" stands for when the compositional design has three models with an equal number of neurons.

$$F1-score = \frac{1}{|l|}\sum_{l}\left(\frac{2 \times Recall_l \times Precision_l}{Recall_l + Precision_l}\right) \quad (2)$$

$$Precision_l = \frac{TruePositives_l}{TruePositives_l + FalsePositives_l} \quad (3)$$

$$Recall_l = \frac{TruePositives_l}{TruePositives_l + FalseNegatives_l} \quad (4)$$

where l is the label for a classification example, $|l|$ is the number of classes, $TruePositives_l$ is the number of samples for which the model infers the correct label l, $FalsePositives_l$ is the number of samples for which the model infers an incorrect label as the correct label, and $FalseNegatives_l$ is the number of samples for which the model infers a correct label as incorrect.

Also, we found that using a 32-part piecewise linear approximation of the activation function and a 32-bit fixed-point representation of all numbers in the design, with 16 bits reserved for the fraction part, results in minimal performance loss due to loss in precision when compiling the designs to VHDL. Furthermore, the ALMs are computed by fitting the developed VHDL designs onto the hardware of Stratix-V FPGA using Quartus Prime Standard Version 19.1. We discontinue design fitting and call the design unsynthesisable if Quartus takes longer than 24 hours or if the design logic requirements are higher than the device can provide. In such instances, we report the ALM estimate as produced by Quartus but do not report the WCET as it cannot be calculated.

Additionally, the WCET is calculated from the maximum cycle frequency of the slow hardware synthesis model of Quartus for each design as shown in Equation 5, where f_{max} is the maximum cycle frequency of the design, and N_{cycles} is the number of cycles required for a single execution of the design.

$$WCET = \frac{N_{cycles}}{f_{max}} \quad (5)$$

Finally, we compare the number of ANN computations, i.e. the number of additions and multiplications, and the number of connections encountered during one pass of a set of inputs. We report the percentage reduction in calculations and connections for the compositional design against the monolithic design.

6.1. Hardware Performance

Figure 4 shows the ALM reductions for all datasets considered in this work. As we can see, the compositional design results in a hardware resource usage (measured using ALM usage) reduction of between 26 and 85 percent, with an average reduction of 53.51 across all datasets. Within the DLC dataset designs, the optimised compositional model and the equally divided compositional model seem to have similar ALM reductions and outperform the three-model compositional model.

Furthermore, we see more variation in the ALM reduction in the Depression dataset as the monolithic and compositional designs do not have the same number of total neurons. Especially for subject ucsd20, the decrease in ALM usage is only 26.7%. This outlier is explained by the massive increase of 187% in the total number of neurons used in the compositional design vis-à-vis the monolithic design, reducing the resource gain. Notably, despite this significant increase in the number of neurons between the designs, the compositional design still needs fewer hardware resources.

6.2. Predictive Performance

Using a compositional design results in a minor reduction in designs' predictive performance for most designs and datasets, as shown in Figure 5. The reduction in $F1-score$ for classification models is less than 0.1 for most designs, with the exception of the 3-model compositional design, where the reduction is around 0.2 for some cases. Within the DLC dataset designs, the optimised compositional model seems to perform the best.

Also, as $F1 - score$ (which lies between 0 and 1) is generally less than 0.1, we can safely say that the reduction is less than 10% for 92% and is less than 5% for 76% of all classification designs. Moreover, the reduction in MAE for regression designs is less than 0.1 for 85% of the subjects. As mood scores are integer values that range between 1 and 7, an absolute error/difference of 0.1 can be considered minor. Interestingly, designs for two subjects (ucsd23 and ucsd29) outperform the monolithic design.

Further, the $F1 - score$ reduction shows a general decreasing trend for most designs for the DLC dataset, with the reduction decreasing as neurons increase. As the neurons increase, the compositional design is able to learn the complex DLC dataset better and begins to catch up with the monolithic design in performance. One exception is the 10-layer design with 118 total neurons, where the compositional design marginally outperforms the $F1-score$ of the monolithic design. Since this is an outlier, we attribute this performance to artefacts arising from randomness in the ANN architecture parameters.

6.3. WCET Performance

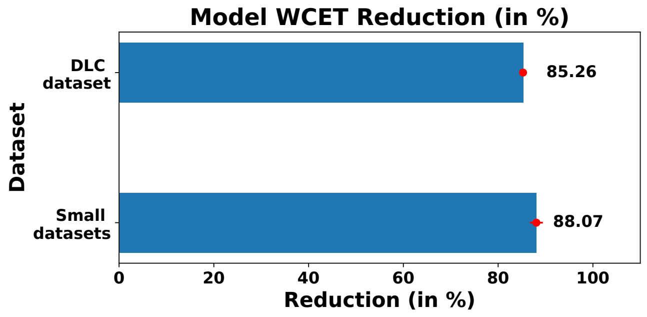

Figure 6: Percentage WCET Reduction for the DLC dataset and the Small Datasets. The DLC and Small dataset reductions are averaged over their respective cases. The red error bars at the end show the variation in the values that have been averaged.

The percentage reduction in WCET for the compositional designs against the monolithic designs for the DLC and Small datasets is shown in Figure 6. We averaged the percentage of WCET reduction across cases for both the DLC and small datasets, as they had minimal variation. We compute and average the WCET reduction for designs that are synthesisable. Overall, 47.22% of the compositional and 25% of the monolithic designs are synthesisable for the DLC dataset designs. All compositional and monolithic designs developed for the Small datasets were synthesisable.

We can see from Figure 6 that the WCET reduction for both the DLC and Small datasets is around 85%. This indicates a significant increase in the network speed for the compositional designs. As the compositional design has a much smaller fan-out of addition and multiplication components on hardware, compositional designs are significantly better in timing performance than monolithic designs.

Furthermore, we found that none of the Depression dataset monolithic designs could be synthesised on hardware due to their high resource requirements. Hence, we could not compute the WCET reduction metric, and the results were not presented. Nevertheless, as all compositional models were synthesisable on hardware, we could compute the WCET of the compositional designs, which was around $1.4\mu s$ for all subjects.

6.4. Computations and Connection Reduction

The percentage reduction in computations and connections for the compositional designs against the monolithic designs for all datasets is shown in Figure 7. We average the values across cases for both the DLC and Depression datasets for brevity. We observe that a compositional design results in an average reduction of computations and connections of 40% over all datasets. This reduction is higher for bigger datasets and larger models at around 40-70%, and at least 24% for smaller datasets.

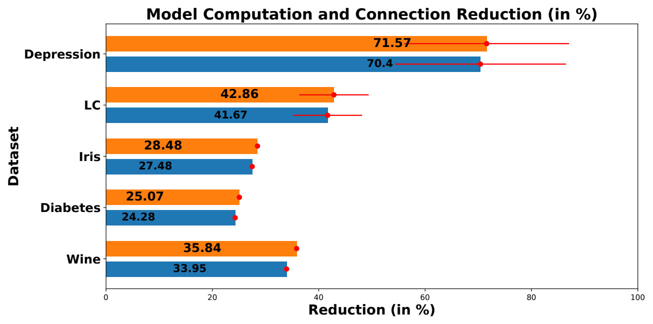

Figure 7: ANN computation and connection reduction for the hardware ANNs for all datasets. The blue bars represent the percentage reduction in computations, whereas the orange bars represent the percentage reduction in connections when using a compositional design instead of a monolithic design. The Red bars show the variation in the values.

The Depression dataset has the highest reduction in computations and connections of around 70% with a standard deviation of around 16%, which can be attributed to a larger number of models per design and unequal neuron numbers between the monolithic and compositional designs. The DLC dataset has a reduction of around 40% with a standard deviation of around 6%. The Diabetes dataset has the lowest reduction in computations and connections, which can be attributed to its small design size.

Also, the reduction in connections and computations is similar across all datasets, as reducing the connections will also reduce the computations, and vice versa. This reduction in connections and computations corroborates the excellent hardware resource gains design we saw when using compositional in previous subsections.

6.5. Summary

Overall, we make the following findings for the MLP-based ANN designs.

- The compositional designs result in a significant reduction in hardware usage vis-à-vis monolithic designs with an average reduction of 53.51% over all datasets at a minor reduction (<5% for most designs) in performance. The reduction in performance decreases as neurons increase.
- The compositional designs result in a massive reduction in WCET vis-à-vis monolithic designs with an average reduction of around 86% over all datasets.
- The compositional designs significantly increase implementable designs vis-à-vis monolithic designs with an average increase of around 58% over all datasets.
- The compositional designs significantly reduce design computations and neuron connections vis-à-vis monolithic designs with an average reduction of around 40% over all datasets.
- The $C-1$ model compositional design performs better at ALM reduction and predictive performance reduction than the C model compositional design for classification problems.

Overall, the smaller size of individual compositional models (through fewer neuron connections) and the reduction in computations in the compositional design should also allow for easier verification of an ANN for its functional specifications. Nevertheless, we emphasise that the compositional designs' performance is intrinsically linked to the architecture chosen for the models and may need some tuning.

7. Conclusion

In this paper, we present a novel application-agnostic approach to replacing single monolithic feedforward ANN designs with compositional designs composed of several smaller ANNs and use several examples and tests to examine the capabilities of each design, both on software and hardware.

We illustrate the approach through a two-stage method to designing compositional ANNs and examine it against monolithic ANNs to discern potential advantages through multiple example classification datasets, including a depression dataset and a Discretionary Lane Changing (DLC) dataset. To aid in proper timing analysis using WCET, we develop a compiler and method for synchronous execution of ANN designs to their hardware equivalents. We further use a multi-cyclic approach that enables the reuse of hardware components.

After a comprehensive comparison, we show that the very nature of compositionality makes it possible to significantly reduce the number of hardware resources (ALMs) required to synthesise the Multilayer Perceptron (MLP) based ANN models into hardware, specifically the Stratix-V FPGA board. Despite the minor performance loss, we discuss how a compositional design helps significantly reduce neuron connections, hardware resources and the WCET. The significant decrease in neuron connections and the overall size of individual models should also make verification easier.

The method presented is not specific to the datasets presented and the type of ANNs considered in this work. We are confident that it can be used with most feedforward ANNs and even combinations of ANNs with machine-learning models with minor to no changes. Moreover, although we have only shown the 2-stage approach with MLP models, where the merge block can be any relevant arbitrary mathematical function, the compositional design allows for arbitrary combinations of models as long as the input-output compatibility of the stages is maintained. We will explore this further in future work. We are also keen on quantifying how a compositional design may aid the functional verification of ANNs.

References

[1] S. Tripakis, "Compositionality in the science of system design," *Proceedings of the IEEE*, vol. 104, no. 5, pp. 960–972, 2016.

[2] R. Alur, *Principles of Cyber-Physical Systems*. The MIT Press, 2015.

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

[4] X. Huang, M. Kwiatkowska, S. Wang, and M. Wu, "Safety verification of deep neural networks," in *Computer Aided Verification*, R. Majumdar and V. Kunčak, Eds. Cham: Springer International Publishing, 2017, pp. 3–29.

[5] E. M. Clarke Jr, O. Grumberg, D. Kroening, D. Peled, and H. Veith, *Model checking*. MIT press, 2018.

[6] T. Mitra, J. Teich, and L. Thiele, "Time-critical systems design: A survey," *IEEE Design & Test*, vol. 35, no. 2, pp. 8–26, 2018.

[7] J. Woodcock, P. G. Larsen, J. Bicarregui, and J. Fitzgerald, "Formal methods: Practice and experience," *ACM computing surveys (CSUR)*, vol. 41, no. 4, p. 19, 2009.

[8] G. Katz, C. Barrett, D. L. Dill, K. Julian, and M. J. Kochenderfer, "Reluplex: An efficient smt solver for verifying deep neural networks," in *Computer Aided Verification*, R. Majumdar and V. Kunčak, Eds. Cham: Springer International Publishing, 2017, pp. 97–117.

[9] X. Yang, P. Roop, H. Pearce, and J. W. Ro, "A compositional approach using keras for neural networks in real-time systems," in *2020 Design, Automation Test in Europe Conference Exhibition (DATE)*, 2020, pp. 1109–1114.

[10] R. Sarić, D. Jokic, N. Beganović, L. Gurbeta Pokvic, and A. Badnjevic, "Fpga-based real-time epileptic seizure classification using artificial neural network," *Biomedical Signal Processing and Control*, vol. 62, p. 102106, 09 2020.

[11] D. Nguyen, H. Ho, D. Bui, and X. Tran, "An efficient hardware implementation of artificial neural network based on stochastic computing," in *2018 5th NAFOSTED Conference on Information and Computer Science (NICS)*, 2018, pp. 237–242.

[12] T. Dreossi, A. Donzé, and S. A. Seshia, "Compositional falsification of cyber-physical systems with machine learning components," 2018.

[13] T. Dreossi, D. J. Fremont, S. Ghosh, E. Kim, H. Ravanbakhsh, M. Vazquez-Chanlatte, and S. Seshia, "Verifai: A toolkit for the design and analysis of artificial intelligence-based systems," *ArXiv*, vol. abs/1902.04245, 2019.

[14] N. Naik and P. Nuzzo, "Robustness contracts for scalable verification of neural network-enabled cyber-physical systems," *Memocode2020*, 2020.

[15] N. Allen, Y. Raje, J. W. Ro, and P. Roop, "A compositional approach for real-time machine learning," in *Proceedings of the 17th ACM-IEEE International Conference on Formal Methods and Models for System Design*, ser. MEMOCODE '19. New York, NY, USA: Association for Computing Machinery, 2019. [Online]. Available: https://doi.org/10.1145/3359986.3361204

[16] S. Gokulanathan, A. Feldsher, A. Malca, C. Barrett, and G. Katz, "Simplifying neural networks using formal verification," 2020.

[17] Y. Y. Elboher, J. Gottschlich, and G. Katz, "An abstraction-based framework for neural network verification," in *Computer Aided Verification*, S. K. Lahiri and C. Wang, Eds. Cham: Springer International Publishing, 2020, pp. 43–65.

[18] N. C. C. f. M. Health (UK), "The classification of depression and depression rating scales/questionnaires," in *Depression in Adults with a Chronic Physical Health Problem: Treatment and Management*. British Psychological Society (UK), 2010. [Online]. Available: https://www.ncbi.nlm.nih.gov/books/NBK82926/

[19] M. K. Ross, T. Tulabandhula, C. C. Bennett, E. Baek, D. Kim, F. Hussain, A. P. Demos, E. Ning, S. A. Langenecker, O. Ajilore, and A. D. Leow, "A Novel Approach to Clustering Accelerometer Data for Application in Passive Predictions of Changes in Depression Severity," *Sensors*, vol. 23, no. 3, p. 1585, Jan. 2023, number: 3 Publisher: Multidisciplinary Digital Publishing Institute. [Online]. Available: https://www.mdpi.com/1424-8220/23/3/1585

[20] R. V. Shah, G. Grennan, M. Zafar-Khan, F. Alim, S. Dey, D. Ramanathan, and J. Mishra, "Personalized machine learning of depressed mood using wearables," *Translational Psychiatry*, vol. 11, no. 1, pp. 1–18, Jun. 2021, number: 1 Publisher: Nature Publishing Group. [Online]. Available: https://www.nature.com/articles/s41398-021-01445-0

[21] E. Balal, R. L. Cheu, T. Gyan-Sarkodie, and J. Miramontes, "Analysis of discretionary lane changing parameters on freeways," *International Journal of Transportation Science and Technology*, vol. 3, no. 3, pp. 277 – 296, 2014. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S2046043016301149

[22] P. Gipps, "A model for the structure of lane-changing decisions," *Transportation Research Part B: Methodological*, vol. 20, no. 5, pp. 403 – 414, 1986. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0191261586900123

[23] D. Xie, Z.-Z. Fang, B. Jia, and Z. He, "A data-driven lane-changing model based on deep learning," *Transportation Research Part C Emerging Technologies*, vol. 106, pp. 41–60, 07 2019.

[24] X. Liu, J. Liang, and B. Xu, "A deep learning method for lane changing situation assessment and decision making," *IEEE Access*, vol. 7, pp. 133 749–133 759, 2019.

[25] R. A. FISHER, "The use of multiple measurements in taxonomic problems," *Annals of Eugenics*, vol. 7, no. 2, pp. 179–188, 1936. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1469-1809.1936.tb02137.x

[26] B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani, "Least angle regression," *The Annals of Statistics*, vol. 32, no. 2, pp. 407–451, 2004. [Online]. Available: http://www.jstor.org/stable/3448465

[27] S. Aeberhard and M. Forina, "Wine," UCI Machine Learning Repository, 1991, DOI: https://doi.org/10.24432/C5PC7J.

[28] S. Chatterjee, J. Mishra, F. Sundram, and P. Roop, "Towards personalised mood prediction and explanation for depression from biophysical data," *Sensors*, vol. 24, no. 1, 2024. [Online]. Available: https://www.mdpi.com/1424-8220/24/1/164

[29] U. D. of Transportation Federal Highway Administration, "Next generation simulation (ngsim) vehicle trajectories and supporting data [dataset]." Accessed 2021-07-09 from http://doi.org/10.21949/1504477, 2016.

[30] Q. Wang, Z. Li, and L. Li, "Investigation of discretionary lane-change characteristics using next-generation simulation data sets," *Journal of Intelligent Transportation Systems*, vol. 18, 06 2014.

[31] J. Zheng, K. Suzuki, and M. Fujita, "Predicting driver's lane-changing decisions using a neural network model," *Simulation Modelling Practice and Theory*, vol. 42, pp. 73 – 83, 2014. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1569190X13001792

[32] J. Močkus, "On bayesian methods for seeking the extremum," in *Optimization Techniques IFIP Technical Conference Novosibirsk, July 1–7, 1974*, G. I. Marchuk, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 1975, pp. 400–404.

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

[34] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc., 2019, pp. 8024–8035. [Online]. Available: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf