Journal of Renewable and Sustainable Energy ARTICLE scitation.org/journal/rse

Short-term solar PV forecasting using computer vision: The search for optimal CNN architectures for incorporating sky images and PV generation history

Cite as: J. Renewable Sustainable Energy **11**, 066102 (2019); doi: 10.1063/1.5122796
Submitted: 31 July 2019 · Accepted: 8 October 2019 ·
Published Online: 12 November 2019

View Online Export Citation CrossMark

Vignesh Venugopal, Yuchi Sun, and Adam R. Brandt[a)]

AFFILIATIONS

Stanford University, Stanford, California 94305, USA

Note: This paper is part of the Special Collection on Best Practices in Renewable Energy Resourcing and Integration.
[a)]**Author to whom correspondence should be addressed:** abrandt@stanford.edu

ABSTRACT

Cloud movement makes short-term forecasting of solar photovoltaic (PV) panel output challenging. A better PV forecast can realize value for both grid operators and commercial or industrial customers with solar assets. In this study, we build convolutional neural network (CNN) based models to forecast power output from PV panels 15 min into the future. Model inputs are the PV power output history and ground-based sky images for the past 15 min. The key challenge is ensuring that due importance is given to each type of input. We systematically explore 28 methods of "fusing" these heterogeneous inputs in our CNN. These methods of fusion (MoF) belong to 4 families. We also systematically explore the many hyperparameters related to model training and tuning. Limited resources preclude an exhaustive search. We apply a three-stage "funnel" approach instead, wherein we narrow our search to the most promising one of these 28 MoF. We find that a two-step autoregression-CNN MoF has the best performance followed closely by a "mix-in" MoF that performs feature expansion and reduction to give appropriate importance to the two types of inputs. The two-step autoregression-CNN model has a forecast skill (FS) of 17.1% relative to smart persistence on the test set comprising 20 complete days (9 sunny, FS = 22%; 11 cloudy, FS = 16.9%). This optimization results in the improvement of FS from 14.1% for a previously published nonoptimized "baseline" model, a CNN wherein the PV history was simply concatenated to the end of the image-sourced vector obtained after convolution, pooling, and flattening operations.

Published under license by AIP Publishing. https://doi.org/10.1063/1.5122796

I. INTRODUCTION

A. Need for short-term solar forecasting

Solar photovoltaic (PV) system output can fluctuate on short time horizons due to cloud passage events. Current electricity systems contain large amounts of dispatchable generation that can be ramped to fill in for the variability of PV under partly cloudy conditions. In future grids with more solar PV, this may continue to be the case, albeit with technologies like storage and demand response mechanisms increasingly performing the function of current dispatchable generation. However, improved PV output forecasts can reduce the uncertainty and help balancing authorities make better operational decisions associated with these resources in real-time power markets. Improved PV forecasts can also help behind-the-meter solar asset owners to strategically manage their local loads to better coincide with the generation from their solar assets. There are benefits to such owners to be accrued through reduction of demand charges, maximization of on-site solar usage, and optimizing the dispatch of any on-site storage/auxiliary generation assets. For this reason, many groups have explored short-term forecasts of PV output.[1–7]

B. Deep learning using heterogeneous inputs

Our previous work used sky images taken from a video camera installed on the roof of the Green Earth Sciences Building at Stanford University and PV panel outputs from a 30.1 kW rooftop system situated 125 m away from the camera. We first built and trained a convolutional neural network (CNN), named the SUNSET (Stanford

University Neural Network for Solar Electricity Trend) for "now-casting," i.e., given a sky image, predicting the contemporaneous PV output.[8] We then used SUNSET to forecast PV output over a 15-min horizon.[9] Given ground-based sky images for the past 15 min on minute resolutions (times t_0 to $t_0 - 15$) and the contemporaneous PV panel output for the same period (henceforth referred to as PV log), SUNSET predicts the power output from the PV panel at times $t_0 + 15$ or 15 min in the future. A pictorial representation of the problem formulation is presented in Fig. 1. This previous work[9] can be consulted for the details on the data processing and general CNN approach.

In this study, we focus on the challenge of data heterogeneity. The two kinds of inputs to SUNSET—historical images and PV output—are extremely different from each other. Every 64 × 64 image with 3 color channels is composed of 12 288 numbers whose value ranges from 0 to 255. Images also naturally have a spatial element inherent in them. On the other hand, power output for the same time-instant is merely a single number that lies between 0 and the peak output of the PV array of interest. As shown in Fig. 1, the fact that we use both these inputs for 16 time-steps (present instant "t_0" to time "$t_0 - 15$" min) adds to the challenges. Making the deep learning system "pay attention" to a single scalar value (PV output) when also presented with a huge array of numbers (sky image) is nontrivial. Such data integration is an active field of research spanning applications like self-driving,[10] geospatial analytics,[11] and robotics.[12]

C. Paper outline

This paper consists of five sections. First is this introduction to the problem at hand and a brief overview of the motivation behind this study. Next, we present a literature review of architectural choices and hyperparameters that were experimented with in this study. In Sec. III, we delve into the details associated with the experimental design. Section IV presents the results and discussion. We conclude in Sec. V with a summary of this study, its key findings, limitations, and possible future steps.

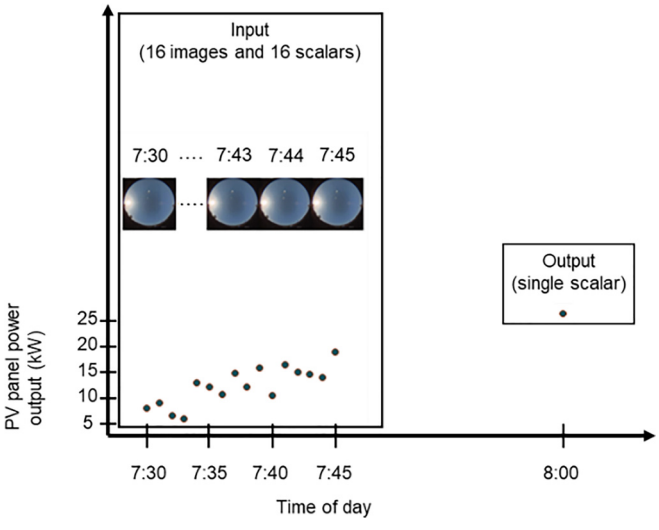

FIG. 1. Illustration of the problem formulation through a specific example. Sky images from 7:30 a.m. to 7:45 a.m. (t_0 – 15 to t_0) on a minutely resolution are to be used in conjunction with contemporaneous PV power outputs to predict the power output at 8:00 a.m. ($t_0 + 15$).

II. LITERATURE REVIEW

We divide the parameters to be explored into 2 subcategories: the method of fusion (MoF) and the hyperparameters. The MoF is the decision around how and where in the architecture, the heterogeneous data inputs are combined. The hyperparameters are model or training settings including the learning rate, regularization strength, neuron activation function, normalization strategy, and convolution and pooling temporal depths. Below, we review previous work in the optimization of these parameters.

A. Methods of fusion (MoF)

Many of the MoF we experimented with are inspired by previous studies in the field of robotics, where image-data fusion is key. For example, an object-grasping robot may be trained using a depth image in conjunction with the *x*-, *y*-, and *z*-position and roll-, pitch-, and yaw-orientation of the gripper to predict a grasp quality score.[13] Besides grasping, several MoF have been tried for robotic tasks like collision avoidance.[12] This study aims to do something similar for solar PV forecasting using sky images and PV log as inputs.

The MoF used in this study can be classified into 4 families: mix-in, activation map combination, activation map stacking, and two-step.

1. Mix-in

The mix-in[12] family of MoF appends nongraphical inputs to the processed images after their spatial characteristics have been processed (Fig. 2). First, the image-sourced input is processed through convolution and pooling. The resulting tensor is then flattened. The flattened vector may then be processed through one or more fully connected layers (FCLs). Nongraphical inputs (either raw or processed through FCLs) are concatenated to the end of this vector. The resulting hybrid vector is then processed further through FCLs. An advantage of this idea is that some dimensionality reduction can be conducted on the processed image, and some feature expansion can be conducted on the nongraphical inputs, both via FCLs. This can better ensure that the information from the latter does not get lost in the sea of information from the former. Also, as described in Ref. 14, the approach of learning completely separate low-level features has proven to be effective when input data modalities differ significantly, for example, with audio and video,[15] and with text and images.[16] The previous SUNSET architecture[9] is a mix-in architecture.

2. Activation map combination

Activation map combination[12] MoF (Fig. 3, left) involves reshaping the nongraphical input to allow direct combination with the image, while spatial characteristics of the image are intact. The vector of nongraphical inputs is processed through FCLs. The resulting vector is then reshaped (and repeated if necessary) into an array. The dimensions of this array exactly match those of the processed image. The 2

Published under license by AIP Publishing

arrays are then combined, through either elementwise addition or multiplication. For example, if the processed image has dimensions of $h \times w \times c$ and there is a vector with n nongraphical inputs to be used with that image, the n nongraphical inputs can be passed through a FCL with c neurons, which will lead to a vector of length c. This vector of length c is then repeated spatially to yield a 3-D array of size $h \times w \times c$ to be combined with the processed image. Alternatively, the raw image can be directly scaled with the corresponding nongraphical input. The assumption behind activation map combination methods is that nongraphical inputs modulate the image in a hierarchical way and determine its relevance.[12] In our problem, we can think of it as modifying (brightening or darkening) a sky image based on the power output from PV panels. A clear sky may produce different amounts of power, dependent on the condition of the panels. If the power output from the panel is low, say, due to high temperature, shading, or a mechanical failure, then having a clear sky will not lead to as high an output as expected.

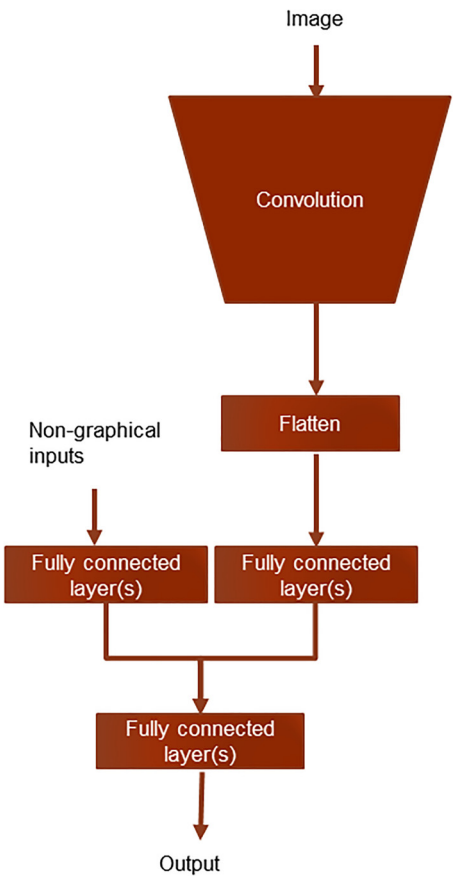

FIG. 2. Illustration of the mix-in family of MoF Images is processed through convolution and pooling. The resulting array is then flattened into a vector. The nongraphical inputs, either raw or after processing through FCLs, are appended to the previously mentioned vector. The augmented vector, which then contains both features from the images and the nongraphical inputs, is further processed through FCLs.

3. Activation map stacking

In activation map stacking MoF,[12] (Fig. 3, right), the vector of nongraphical inputs, either raw or processed through FCLs, is repeated and reshaped to the spatial dimensions of the raw image or to the processed image after several layers of convolution and pooling. The two arrays are then stacked onto each other along the channel dimension. If the nongraphical data inputs are stacked onto the raw image, the image now has more than 3 channels (i.e., R, G, B, and one or more data channels). After stacking, further convolution is performed on the stack.

4. Two-step

In the two-step MoF, illustrated in Fig. 4, the forecasting task is split into two steps. In the first step, persistence or autoregression is used with the PV log as input. In the second step, a CNN-based model[9] is trained to predict the error in the first step's prediction. Assume that the true PV output at $t_0 + 15$ is Y. The first model, say an autoregression model, uses the PV log to make a prediction of PV output of Y', with the error in prediction being $Y - Y' = \varepsilon$. The second model is a CNN which uses images. It is trained to predict ε. Its prediction of the error, ε', is added to Y' to yield the combined model prediction. Motivation for the two-step approach is in how a human might approach the task, who might look at inputs sequentially. For example, you look at PV log, see an increasing trend and thus forecast that the output will continue to increase. But then, you look at the sky images and see that a cloud is approaching the position of the sun, and based on this, revise your initial prediction downward.

By explicitly separating the heterogeneous data sources in sequential prediction processes, two-step approaches may make it easier for the deep learning system to account for the importance of the PV log.

B. Hyperparameter tuning

To make a CNN-based model, in addition to choosing a MoF, we must tune various model settings, called hyperparameters, which control how and what the model learns. Training occurs by optimizing the model weights and biases to minimize the L2 loss function, i.e., the prediction mean squared error (MSE). The Adam optimizer[17] is used here given its merits including but not limited to: computational efficiency, suitability for high dimensionality, nonstationary objectives, and ability to use noisy/sparse gradients. The learning rate, a measure of the step-size in every Adam update, is a hyperparameter to be tuned.

Given the large number of weights and biases in a deep learning model, overfitting is a concern. Dropout is a regularization technique that addresses overfitting by randomly dropping some neurons and associated connections during training.[18] The percentage of neurons to be dropped is a hyperparameter to be tuned.

When input features are extremely different, it is common to z-normalize every feature. With CNNs, the distributions of each convolutional layer's inputs can change too. Batch normalization[19] layers are widely used in practice to address this. Since batch normalization involves computing the mean and standard deviation for every channel across all examples in the mini-batch, its effect is dependent on the mini-batch size. Layer normalization[20] and instance normalization[21]

Published under license by AIP Publishing

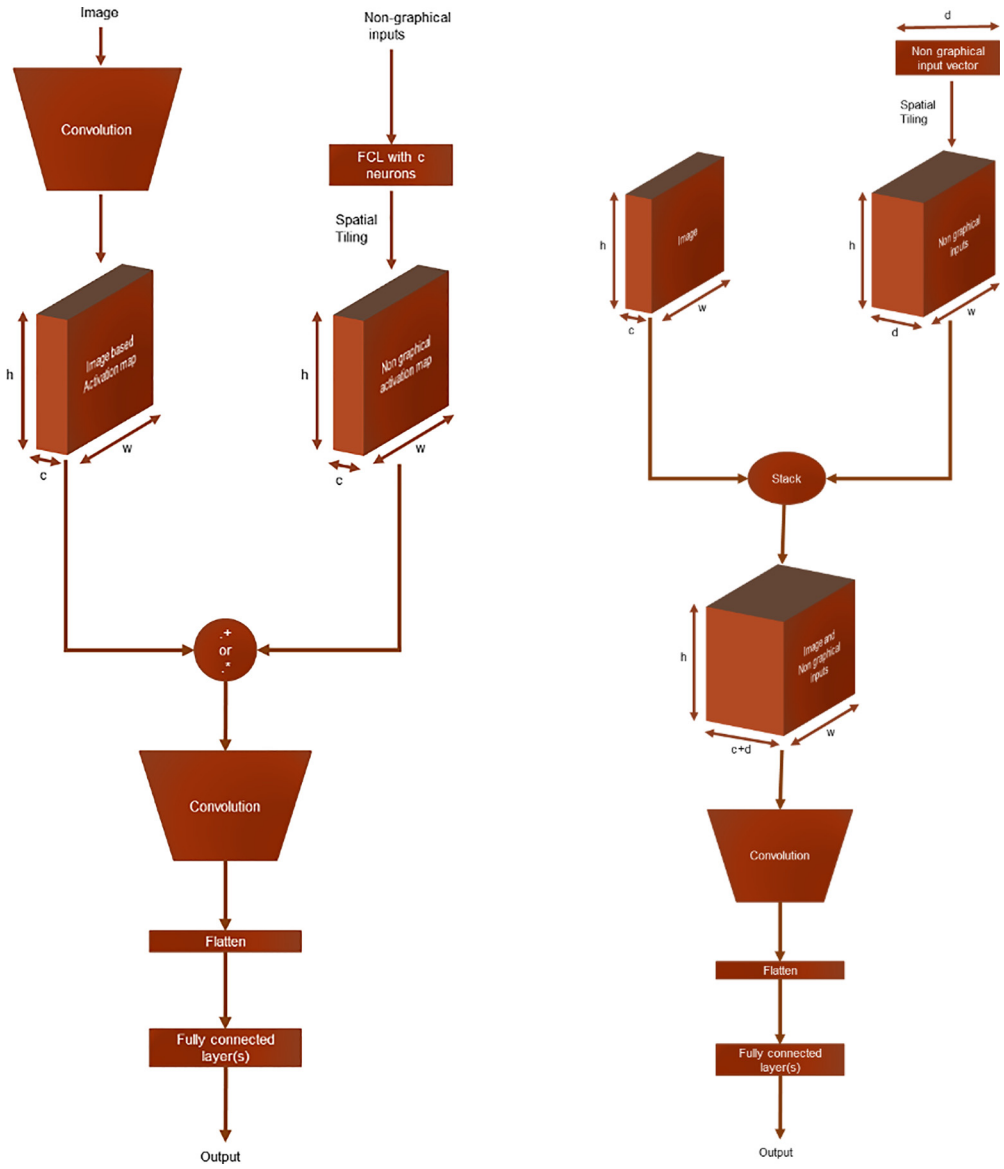

FIG. 3. Illustration of the activation map combination (left) and activation map stacking (right) families of MoF. In contrast to the mix-in MoF, the nongraphical inputs are combined with the image, while the latter's spatial characteristics persist. This is achieved by processing nongraphical inputs via FCLs and then reshaping and repeating them if necessary, to coincide with the dimensions of the image-based array to allow for fusion via elementwise operations or stacking.

are alternative strategies that get rid of this dependence and ensure the same behavior during training and inference. Finally, there is the option of having no normalization layers in the architecture. We explore all these normalization options as discrete choices to be sampled from in each experiment.

Activation functions are used to introduce nonlinearity in the response of the model to the inputs. The Rectified Linear Unit (ReLU) activation has been found to improve CNN performance[22] and is a common choice. In the absence of any normalization within the architecture, we use a close variant named leaky ReLU which has a small slope in the negative domain. This brings us closer to zero-mean outputs and avoids the issue of having a negative saturation regime.[23] The slope in the negative domain of leaky ReLU is a hyperparameter to be tuned. The choice of normalization strategy and activation function mentioned here is solely for the convolution layers. Since we do not use any normalization between FCLs, the activation function for the neurons in those layers is always ReLU.

Since we are using information from present instant t_0 to time $t_0 - 15$ min, there are both spatial and temporal characteristics in the inputs. To capture these spatiotemporal characteristics, we experiment

Published under license by AIP Publishing

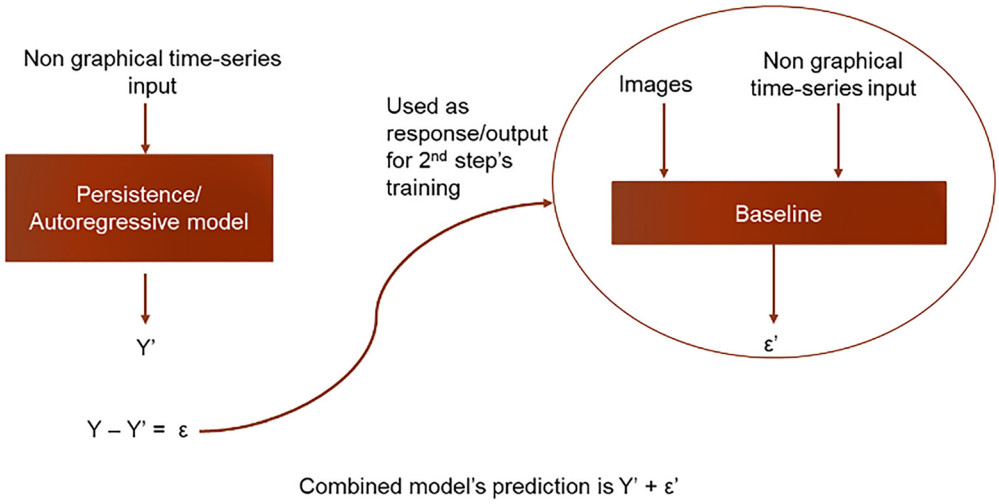

FIG. 4. Illustration of the 2-step family of MoF. The first step involves using nongraphical inputs alone to predict the response variable. The second step involves making use of the image-based input to predict the error in the first step model's prediction. The combination of the predictions from the two steps gives the final prediction of the response variable.

with 3D convolution and pooling[24] in addition to the 2D variants. 2D convolution helps the model learn spatial characteristics such as edges, shapes, and color contrasts present in an input image. When the input consists of a stack of images or video frames, 3D convolution builds on the idea of 2D convolution in the third, temporal dimension, to capture the movement of the spatial features with time (Fig. 5).

III. METHODOLOGY

A. Data

Training CNNs is time intensive. The fact that we have many hyperparameter choices and thus experiments to conduct makes using the entire dataset during training impractical. To expedite the process, we use a subset of the dataset for most of the experimentation. This subset is still spread across the entire year to cover all weather conditions. Once we end the search process, we retrain the best architectures on the entirety of the dataset. The size of the dataset used is given in Table I. The test set consists of 20 days that were never used in any training, model tuning, or decision-making. Nine of these test days are sunny and the rest are cloudy, picked based on visual inspection of the data. For more details about the dataset, refer to.[9]

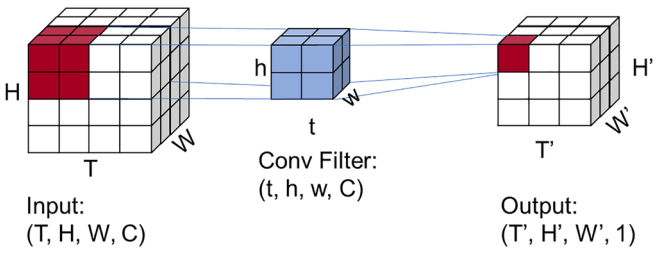

FIG. 5. Illustration of 3D convolution. A convolution filter parses through the entire image computing dot-products with small portions of the image at a time. The dimensions of each of these portions exactly match the dimensions of the filter. A "2D" filter looks for spatial characteristics such as edges, shapes, and color-contrasts in the image. A "3D" filter in addition to that, also acts across the temporal dimension of a stack of images to track the movement of the identified spatial characteristics with time. In our experimentation, the spatial dimensions of the convolution filter, "h" and "w," are always equal to 3 each. The temporal depth, "t," is sampled as shown in Table I. The filter always spans across all channels of the image.

B. The measure of success

This study aims to find the best CNN-based architecture(s) for using the PV log and sky image history to predict future PV output. The measure of success is the five-fold cross validation (CV) loss. The *k*-fold CV approach gives a much better estimate for the test error as compared to the validation set or leaves one out cross validation approach as detailed in Sec. 5.1.4 of Ref. 25.

TABLE I. Size of the dataset used in this study. Each sample consists of 3 items: (1) sixteen 64 × 64 × 3 sky images, a minute apart from each other, (2) sixteen (minutely averaged) PV power outputs corresponding to the minutes that the sky images were taken in, and (3) PV power output 15 min into the future to be used as the response variable that the model will be trained to predict.

Purpose	Training and validation samples	Testing samples
Search for best MoF and hyperparameter tuning (stages 1, 2, and 3)	8732	NA
Final retraining and testing of best architectures	68833	8075

Published under license by AIP Publishing

TABLE II. Families of MoF, references, names, and descriptions of (inspired) MoF in this study that will be referred to later.

MoF family	References	MoF in this study belonging to this family	
Mix-in	9, 12, 14, 28, 29, 30, 31, 32, 33, 34, 35, 36, and 37	Mix_0FC (baseline) Mix_Img_2FC Mix_PV_1FC_Img_1FC (Y-model)	Images are processed through convolution-pooling (CP) layers and then flattened, leading to an image-sourced vector (Img). This vector is passed through a certain number of fully connected (FC) layers. PV log is passed through a certain number of FC layers as well. The outputs from these 2 sets of FC layers are then concatenated horizontally (mixed) and fed into more FC layers.
		Mix_PV_1FC(n1)_Both_1FC(n2)_Img_1FC(n3) (Yuan-Model_n1_n2_n3)	In addition to a FC layer, each for PV log and Img, there is one FC layer, which is fed with both PV log and Img. These FC layers have n1, n3, and n2 neurons, respectively. The output from these 3 FC layers are concatenated and fed into a subsequent FC layer.
Activation map combination	12, 38, 39, and 40	Channelwise_add/mul_1FC_1CP Channelwise_add/mul_2FC_1CP Channelwise_add/mul_1FC_2CP Channelwise_add/mul_2FC_2CP	Images are processed through a certain number of convolution-pooling (CP) layers. PV log is processed through a certain number of FC layers. The number of neurons in the FC layers corresponds to the number of convolution filters (and thus channels of output tensor). 24 filters are used in the first C layer and 48 in the second. The output from the latest FC layer is repeated so that it matches the spatial dimensions of the output from the latest CP layer. The two are then combined through elementwise addition/multiplication.
		Spatial_add/mul_1FC_1CP Spatial_add/mul_1FC_2CP	Only difference from above-the number of neurons in the FC layers is the product of the spatial dimensions of the output from the latest CP layer, 1024 (32×32) after 1 CP and 256 (16×16) after 2 CP. This spatial downsizing occurs due to 2×2 max pooling. The output from the latest FC layer is repeated across the channel dimension so that it matches the dimensions of the output from the latest CP layer. The two are then combined through elementwise addition/product.
		Image_modulate_1FC	PV log data are normalized to lie in the range of 0–1, with 1 corresponding to the peak-output of the PV panels. The raw sky images are then elementwise multiplied with the corresponding (normalized) PV log value.
Activation map stacking	12, 41, 42, and 43	Channel_stack_0FC_1CP Channel_stack_1FC_1CP Channel_stack_0FC_0CP	Sky images are processed through a certain number of CP layers. PV log is processed through a certain number of FC layers. The output from the FC layers is then repeated spatially to match the spatial dimensions of the output from the CP layers. The 2 tensors are then concatenated (stacked) across the channel dimension.
Two-step	…	2_step_AR(16) 2_step_Naive_Persistence 2_step_Smart_Persistence 2_step_Smart_Persistence_Geom	In the 1st step, PV log alone is used for prediction using a persistence-based or autoregressive (AR) model. In the 2nd step, both sky images and PV log are used in a CNN with the baseline MoF to predict the error in the 1st step's prediction.

C. Identifying methods of fusion and hyperparameter choices

We experiment with 28 MoF belonging to 4 families mentioned in Table II. The MoF are named to reflect how the two data inputs are integrated, followed by where in the architecture, the integration takes place. For example, in Mix_PV_1FC_Img_1FC, the data inputs are integrated through vector concatenation as is the case with the mix-in family of MoF. Before fusion takes place, the PV log is passed through 1 fully connected (FC) layer. The sky images [after being processed

Published under license by AIP Publishing

through convolution-pooling (CP) layers] are passed through 1 FC layer as well. If fusion takes place in between convolution-pooling (CP) steps, then the number of CP steps that the images have passed through before fusion is also mentioned in the name.

Across our MoF, we have 7 hyperparameters, 3 of which are continuous and the rest, categorical. The hyperparameters and corresponding choices/ranges are listed in Table III. Leaky ReLU is used as the activation function only when the normalization strategy within the architecture is chosen to be "None." In any other case, the activation function used is ReLU. The convolutional section of the CNN architecture is similar to that used in the SUNSET architecture. One difference is associated with the temporal depth of the convolution and pooling filters. 2D convolution and pooling, as used in SUNSET, lead to filters that span across all time steps. In practice, the image-based array is reshaped to combine the color channel dimension and the time step dimension into one. 3D convolution and pooling need a temporal depth to be defined for both operations. The 2 numbers present within the parentheses are the temporal depths in the two convolution-pooling steps involved in the SUNSET architecture.

D. Sampling strategy

Given the many hyperparameters and respective choices, trying every single combination of hyperparameter settings is impractical. Sampling is thus necessary. Random sampling is employed, given its advantages in high-dimensional spaces.[26] Categorical hyperparameters are sampled using a uniform probability distribution, sampling from categorical options available. Continuous hyperparameters are sampled using the Latin hypercube sampling strategy.[27]

E. Limiting the impact of externalities

To ensure a fair comparison between the different architectures, we set a fixed seed during dataset division to ensure that the training and validation sets are divided consistently across all experiments. However, since graphical processing units (GPUs) are used to train these models, some noise cannot be removed. Different cores in a GPU may finish computations at different times and floating-point operations are not always associative.

TABLE III. Hyperparameters and corresponding stage-1 sampling choices.

Hyperparameter	Sampling range/ categorical choices
Learning rate	5×10^{-7} to 1×10^{-5}
Dropout rate	10% to 50%
Slope for leaky ReLU	0.01 to 0.1
Z-normalization of inputs and outputs	Yes, no
Normalization strategy within architecture	Batch, layer, instance, none
Convolution type and temporal depth	2D 3D: (1, 1), (3, 3), (5, 5), (7, 7), (3, 5), (5, 3), (1, 3), (3, 1), (5, 7), (7, 5)
Pooling type and temporal depth	2D 3D: (1, 1), (2, 2), (1, 2)

F. Experimental setup

We have an 8-dimensional search space in this study. The MoF lies on one of the dimensions, and the 7 hyperparameters lie on the rest. We start with a broad search over this 8-dimensional space and then sample more finely in later stages, focusing on promising sections of the search space which have good initial results. The experiments in this study are conducted over three stages. Figure 6 shows this approach as a "funnel" experimental design. In every experiment, a single set of randomly sampled hyperparameters are used in conjunction with every MoF retained in that stage.

To gauge if we have explored the search space in a given stage enough to curtail it, we look at the marginal benefit of conducting more experiments. We track the best performance to date as a function of the number of experiments performed. After a certain point, conducting more experiments will not yield a much better architecture than that we have already found. We then end that stage, curtail our choices in MoF and hyperparameters, and proceed to the next stage.

Our search ends with the 3rd stage. 4 architectures remain, each with a different MoF. These 4 architectures are finally retrained with a larger dataset, as mentioned in Table I. We also retrain and retest the SUNSET baseline (with hyperparameters chosen in Ref. 9). The use of 3D convolution in some of our 4 final architectures has added many more weights to our architecture, which leads us to be GPU memory constrained. Thus, we can only use a batch-size of 32. Since the SUNSET baseline was originally trained with a batch size of 256, we retrain it with a batch-size of 32 and with a seed consistent with that of the others for a fair comparison.

IV. RESULTS AND DISCUSSION

Experimentation was conducted across 3 stages to find the best MoF and to tune hyperparameters to be used in conjunction with them. An overview of the results is provided in Fig. 7, which shows how the choices (both in terms of MoF and hyperparameters) are curtailed as we transition from one stage to the next. We will now delve deeper into the findings in each of these stages.

A. Stage 1: Broad exploration

The first stage of experimentation was a broad level of search with the widest bounds on hyperparameter settings. In the first stage, we conducted 40 experiments with each of the 28 MoF for a total of 1120 experiments. The results from these experiments are displayed in Fig. 8.

1. Curtailing methods of fusion

Based on Fig. 8, the Yuan-model based MoF seems to be promising, so do some of the others like 2_step_AR(16), Y-model, and Spatial_add_1FC_1CP. On the other hand, there are some MoF like Channelwise_mul, Baseline, and 2_step_Naive/Smart_Persistence that do not perform as well. In moving to stage 2, we retained only 8 of the 28

Published under license by AIP Publishing

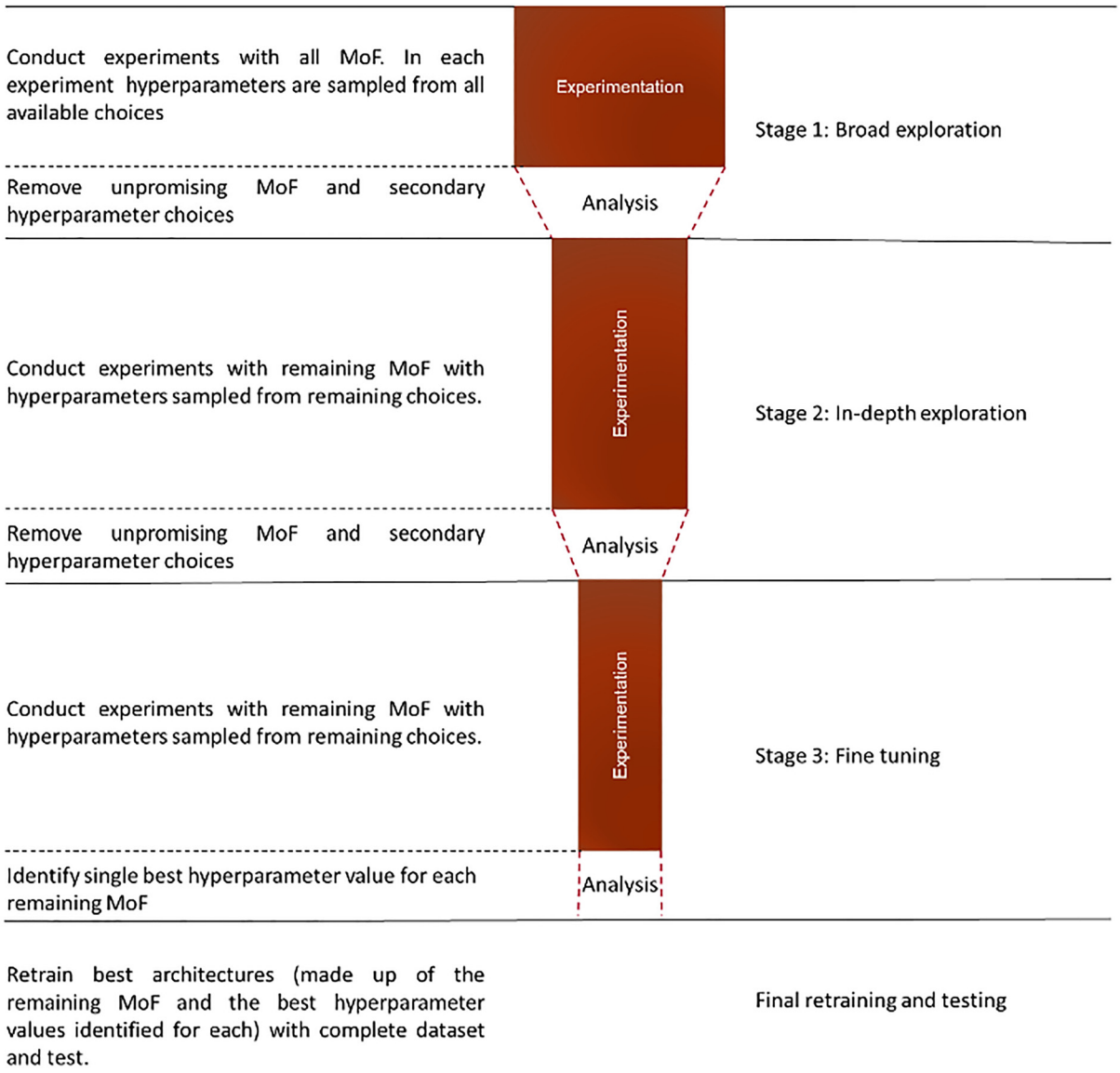

FIG. 6. Illustration of the funnel experimental setup. We start with a broad search across the widest ranges of hyperparameter choices and MoF options. In every subsequent stage, only a promising subset of these are retained and further experimented with.

MoF. The MoF to retain are selected using two complementary metrics. First, we count the number of experiments out of 40, wherein the novel MoF outperformed the SUNSET model,[9] retrained with the same dataset and seed. Second, we ordered performance across the 40 experiments for each MoF, filtering for the lowest CV Root Mean Square Errror (RMSE) for each MoF. The top 8 scoring MoF in metric 1 were also the top 8 scoring in metric 2. So these 8 MoF were retained for the second stage.

2. Narrowing range of hyperparameter settings

In addition to removing MoF, we also aim to curtail hyperparameter choices to sample from. To give an example, we will show the process for narrowing the sampled dropout rate. Figure 9 shows the fivefold CV RMSE attained in the 40 stage 1 experiments as a function of the dropout rate used, for the Y-model. The top 5 performances (meaning 5 lowest CV RMSE) are emphasized in bright red, and the 35 inferior performances are shown in faint red. Once the 5-best dropout rates are identified, the lower bound of the dropout rate for the next stage of experimentation is set to 10% lower than the smallest of these 5 values. Similarly, the upper bound is set to 10% higher than the highest of the 5 best-performing dropout rates. For our 8 retained MoF, this process gives us 8 ranges of dropout rates that worked well, one for each MoF to be retained. The union of these 8 ranges then becomes the sampling range for

Published under license by AIP Publishing

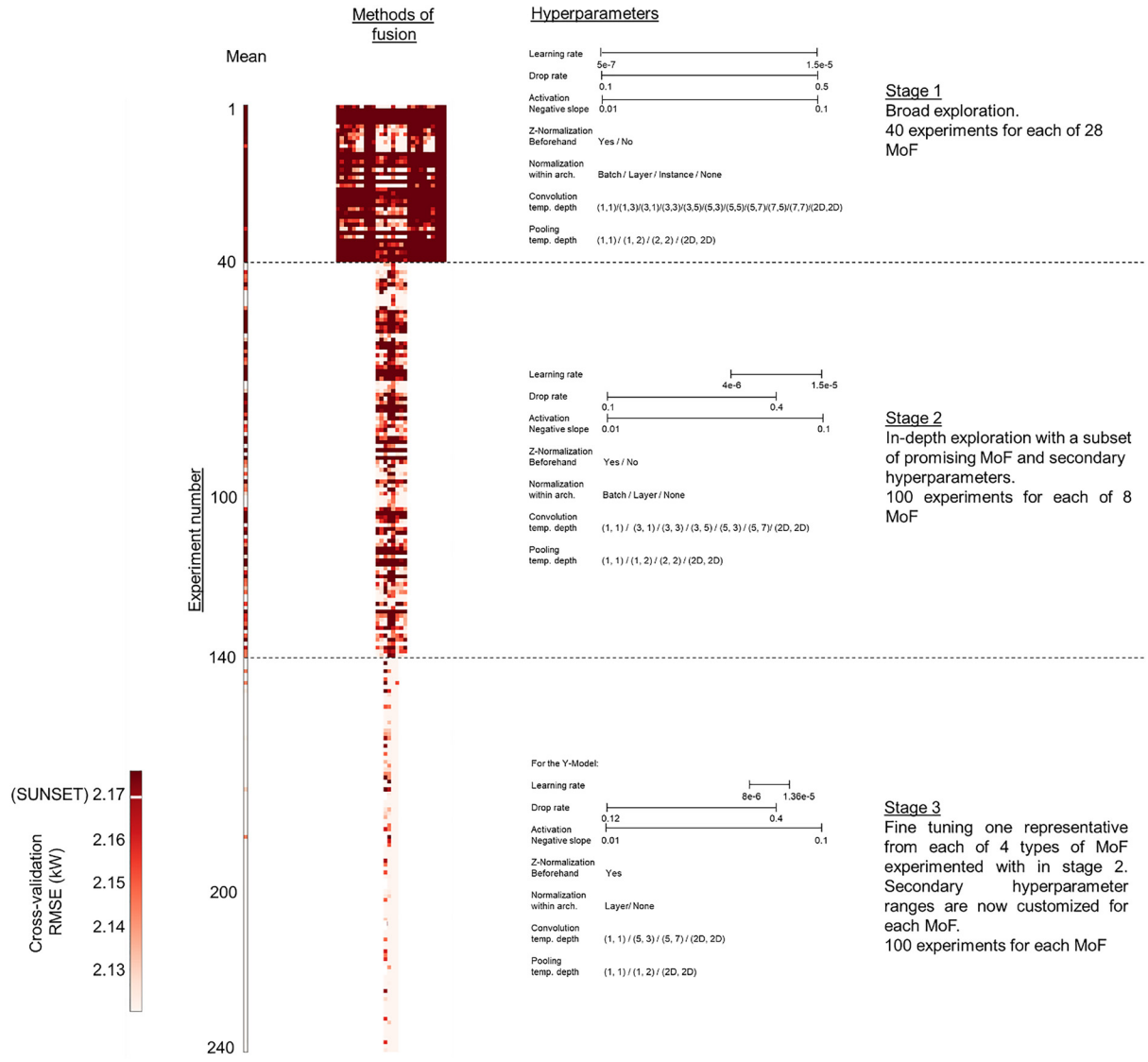

FIG. 7. Overview of the results from the 3 stages of experimentation spanning across 240 experiments in total. The fivefold CV RMSE attained by the different MoF in the experiments conducted as part of each stage is represented as a heat map on the left. The hyperparameter choices available to be sampled from in each experiment for one MoF in specific, the "Y-model," are listed on the right. The increasing fraction of lighter-colored pixels from one stage to the next denotes improving performance as we narrow down to the most promising of the MoF and hyperparameter choices.

the dropout rate in the 2nd stage of experimentation. This analysis is repeated for the rest of the hyperparameters.

B. Stage 2: In-depth exploration

In stage 2, we conducted 100 experiments with each of the 8 MoF retained from stage 1. The CV RMSE attained in these experiments is shown in Fig. 10. We looked at the marginal benefit of conducting more experiments while deciding when to terminate stage 2 of experimentation. In Fig. 11, we see how the lowest CV RMSE attained by a MoF changes with every 5 additional experiments, in the order that the experiments were performed. With only a few exceptions, the best performance has been more or less stabilized by the end of 100 stage-2 experiments. The black dashed lines indicate the transition from one stage to another. A sudden improvement in performance at the stage transition is expected (and desired) since the hyperparameter ranges/choices are curtailed to only include promising options.

1. Curtailing methods of fusion

Based on Fig. 10, the Yuan-model based MoF again seems to be performing consistently. However, it is important to note that

Published under license by AIP Publishing

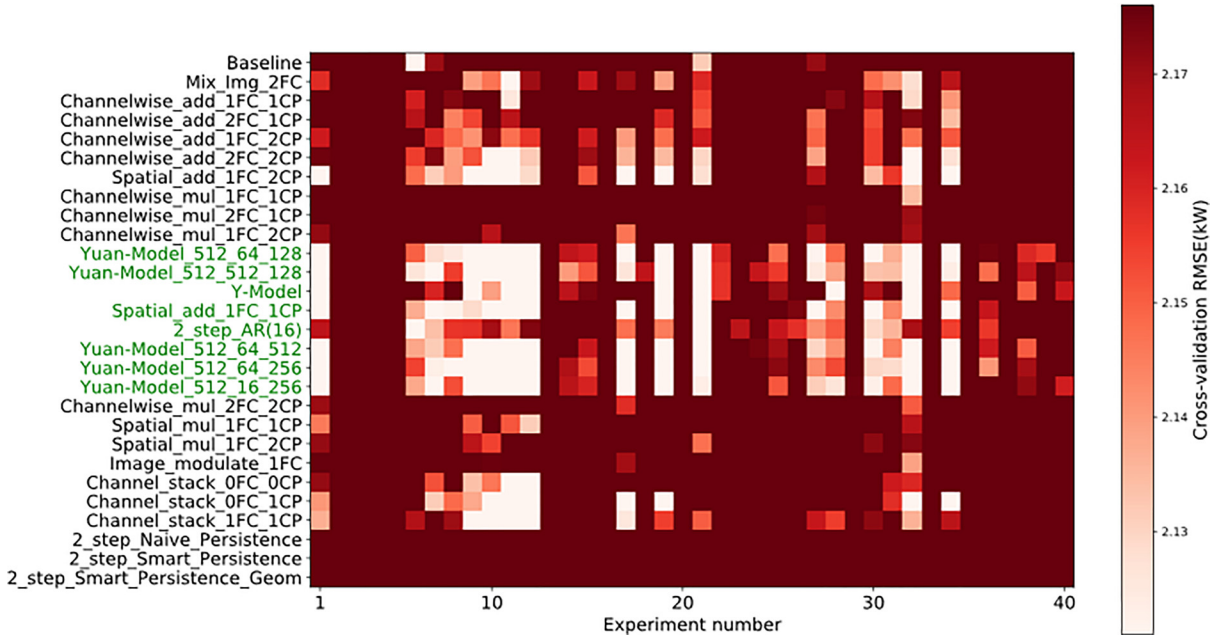

FIG. 8. Cross-validation RMSE at the end of the 1st stage of experimentation represented as a heat map. The MoF retained for the 2nd stage are emphasized in green.

the peak performances of each of the 8 MoF are not significantly different as evident in Fig. 11. Since 5 of these 8 MoF are all Yuan-model-based, we decided to retain only 1 of these at the end of stage 2. Yuan-model 512_64_512 was found to be consistently better than the rest of the Yuan-model variants and thus retained for the 3rd stage.

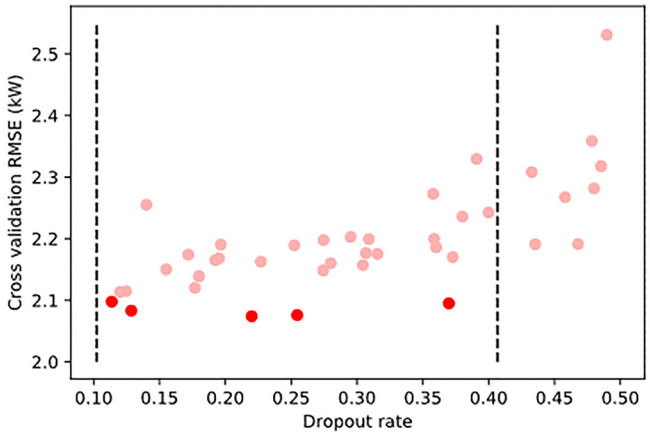

FIG. 9. Cross-validation RMSE as a function of the dropout rate for the Y-Model at the end of the first stage of experimentation. The bright red markers correspond to the five lowest CV RMSE. The black vertical lines mark the boundaries of the sampling range for the next stage of experimentation. The boundaries are 10% lower than the lowest of 5 best dropout rates and 10% higher than the largest of 5 best dropout rates.

2. Narrowing range of hyperparameter settings for each method of fusion

As we did before, we identified the hyperparameter choices that led to the 5 best performances for each of the four MoF chosen to be retained. But in contrast to the analysis at the end of stage 1, we refrained from taking the union of the 4 ranges (one from each of the MoF retained). The secondary hyperparameters for each of the 4 MoF were curtailed based on what worked for that MoF and not any other. This is important as we are narrowing in on the optimal settings for each MoF. For example, the ideal learning rate range for the Y-model was found to be 8×10^{-6} to 1.35×10^{-5}, while that for the Spatial_add_1FC_1CP was 3.6×10^{-6} to 6.5×10^{-6}. So taking the union of these ranges for experimenting in the 3rd stage would clearly be suboptimal.

C. Stage 3: Fine-tuning

The aim of the third stage was to find the single best set of hyperparameters for each of the 4 MoF that made it thus far. In stage 3, we conducted 100 more experiments with each of these 4 MoF. The CV RMSE attained in these 100 experiments is shown in Fig. 12. To decide if we can terminate the 3rd stage, we again looked at the marginal benefit of conducting more experiments (Fig. 11). The best performance for the 4 MoF has more or less stabilized with one exception, for 2_step_AR(16). We thus terminated our search at 100 experiments. We then had four architectures—each with a different MoF and the corresponding hyperparameters that worked best in conjunction with that MoF. The final step of experimentation and this study was to retrain these 4 architectures on a

Published under license by AIP Publishing

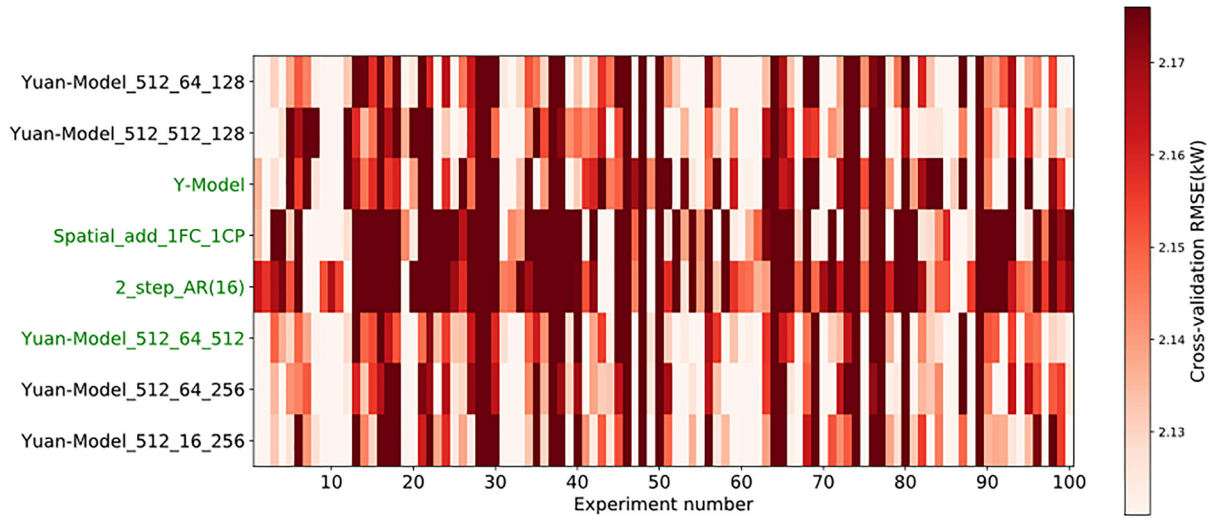

FIG. 10. Cross-validation RMSE in the 2nd stage of experimentation represented as a heat map. The MoF retained for the 3rd stage are emphasized in green.

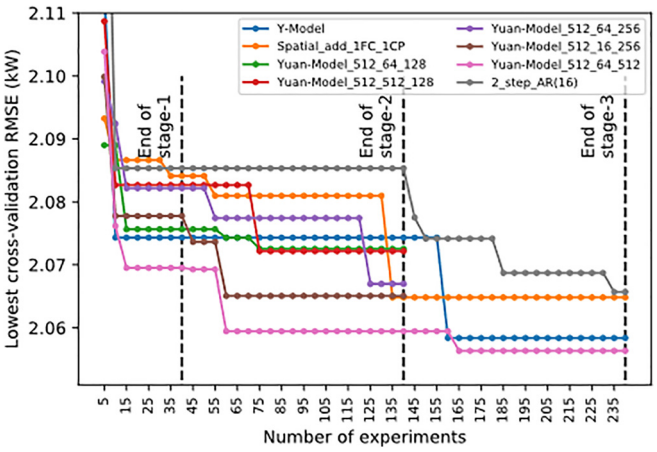

FIG. 11. Marginal benefit of every 5 additional experiments expressed by the lowest CV RMSE attained by a MoF as a function of the number of experiments, in the order performed. The black vertical lines mark the end of the 1st, 2nd, and 3rd stages of experimentation.

bigger dataset and finally use them for inference on the never-before-used test set.

D. Retraining with a larger dataset and testing

The final stage involved retraining and testing the four best architectures. We used the entirety of the dataset, as mentioned in Table I. The process was repeated for the SUNSET baseline[9] for comparison. To check if the difference in the performance-level of different MoF is significant, we conducted the experiment 5 times, each time with a different seed.

The cross-validation and test RMSE (overall, sunny alone, and cloudy alone) across the 5 reruns are shown in Figs. 13 and 14. The mean RMSE across these 5 reruns is listed in Table IV. Of the final four MoF, 2_step_AR(16) and Yuan-model_512_64_512 perform the best and have a forecast skill of 17.1% and 16.7% relative to smart persistence on the overall test set. Smart persistence is hard to beat on sunny days, with only Yuan_512_64_512 and 2_step_AR(16) outperforming it on sunny days. However, smart persistence is significantly outperformed by all MoF on cloudy days.

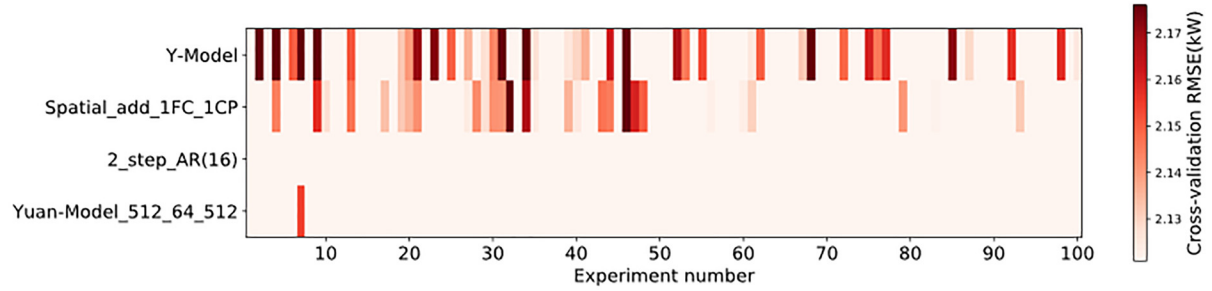

FIG. 12. Cross-validation RMSE in the third stage of experimentation represented as a heat map. The superiority of the MoF and hyperparameters retained in this stage is evident in the low CV RMSE relative to the results corresponding to the first and second stages.

Published under license by AIP Publishing

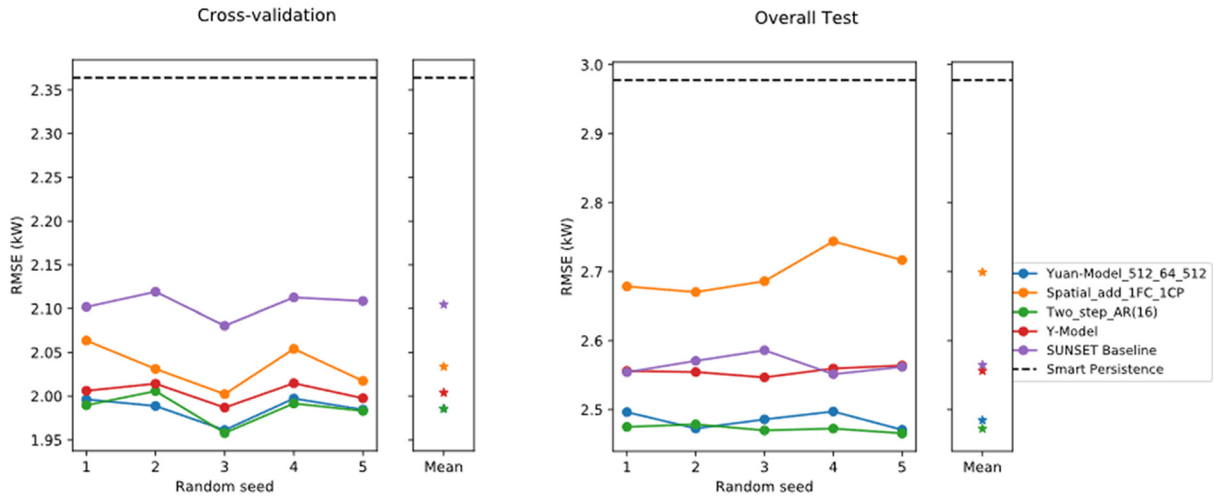

FIG. 13. Cross validation and overall test set performance of the 4 best MoF and the SUNSET baseline over 5 retrainings, each with a different seed. All 5 models significantly outperform smart persistence with 2_step_AR(16) and Yuan-model_512_64_512 being the best.

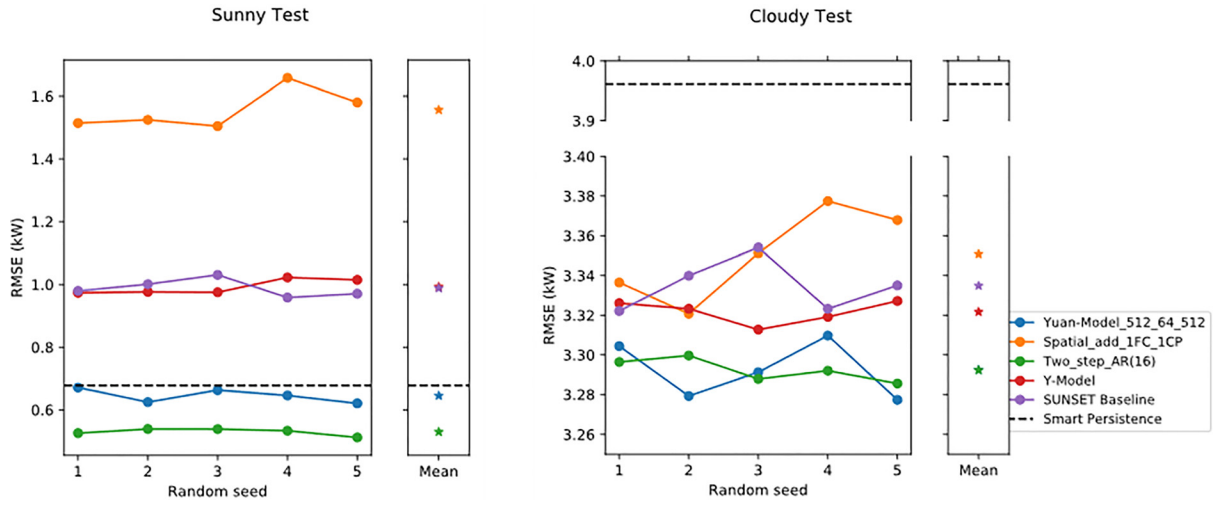

FIG. 14. The overall test performance shown in Fig. 13 is segregated into sunny and cloudy test set performance. Smart persistence is tough to beat on sunny days but is significantly outperformed by all the 5 CNN-based models on cloudy days. 2_step_AR(16) and Yuan-model_512_64_512 perform the best in both sunny and cloudy conditions.

TABLE IV. Summary of the results from the final stage of experimentation. Training, validation, and test performance of the smart persistence and SUNSET baseline are presented in addition to the performance of the 4 architectures tuned and retained at the end of the 3rd stage of experimentation.

Model	Training RMSE (kW)	Validation RMSE (kW)	Sunny test RMSE (kW)	Cloudy test RMSE (kW)	Overall test RMSE (kW)	Forecast skill (overall test) (%)
Smart persistence	2.36	2.36	0.68	3.96	2.98	0%
SUNSET baseline	2.45	2.13	0.98	3.33	2.56	14.09%
Yuan-model_512_64_512	2.04	2.01	0.65	3.29	2.48	16.78%
Spatial_add_1FC_1CP	2.24	2.06	1.56	3.35	2.70	9.39%
2_step_AR(16)	1.90	2.02	0.53	3.29	2.47	17.11%
Y-model	2.18	2.03	0.99	3.32	2.56	14.09%

Published under license by AIP Publishing

V. CONCLUSION

This study aimed to find the best CNN architecture(s) for combining the merits of sky images and PV log from the past 15 min on a minutely resolution for a 15 min ahead solar PV forecast. We experimented with 28 MoF belonging to 4 families and identified through a literature review mostly centered on the field of robotics. We conducted experiments across 3 stages to find the best of these MoF and tuned hyperparameters to be used in conjunction with them.

A. Summary of results

2_step_AR(16) and Yuan-model_512_64_512 MoF were found to be the best. Their overall test forecast skills were 17.11% and 16.78%, respectively, relative to smart persistence. With the Yuan-model, in the very first FCL, some neurons have access to the image-based inputs, some have access to the nongraphical inputs, and some have access to both. As described in Ref. 15, learning completely separate low-level features when input data modalities differ significantly seems to be effective. Some associations can also be captured using common neurons. Using an equivalent number of neurons for both image and PV log alone ensures that the dimensionality reduction of image-sourced features takes place and feature expansion of PV log takes place. This means that 16 PV log values do not get lost in a vast amount of information coming from 16 images. The 2-step model thrives on the idea of first using PV log alone, before bringing sky images into play.

Some interesting findings when it comes to hyperparameter choices are as follows: Increasing the convolution depth beyond 3 time-steps degrades performance. Given the spatial dimensions of the convolution filter are fixed at (3, 3), it is empirically found to be the best to maintain homogeneity and have the temporal depth of 3 as well. This finding echoes that of previous work[24] with supervised video analysis for object and action recognition, scene classification, etc. We also found that it is best to either z-normalize inputs beforehand or use a normalization layer within the architecture but not both.

B. Unanswered questions, limitations, and future work

One of the concerns with deep learning is the worsening of performance with additional inputs. Deep learning models have a lot of free parameters (weights) and thus the flexibility to potentially overfit to the noise and miss out on the helpful signal. The next natural step would be to check if the best MoF identified overcomes this concern by adding in other heterogeneous data types such as weather features like temperature, aerosol and optical depth.

We believe that one of the merits of the Yuan-model, Y-model, and similar MoF lies in the fact that the processed image-sourced features are down-sampled and PV outputs are up-sampled to ensure that the resulting dimensions are comparable. We currently down-sample the image using FCLs. It would be interesting to see if principal component analysis can be used instead. Given that the 2_step_AR model performs well, it is also worth trying other variants in both the steps. Can we get better performance by using, say, ARIMA in the first step and Yuan-model in the 2nd step? In the Yuan-model based MoF, we explicitly set the number of neurons in the FCLs that had access to each of the input sources as also both. The group regularization strategy detailed in Ref. 14 would be an alternative worth trying. Given the temporal nature of our problem, we use 3D convolution and pooling. A natural extension would be to try recurrent neural networks instead.

The computational cost has precluded conducting an exhaustive search across our parameter space. So there is the possibility of having disregarded a MoF before finding the optimal hyperparameters for it. Also, we curtail hyperparameters by looking at the CV RMSE as a function of that hyperparameter alone. This does not account for interactive effects. The best MoF found for the task of short-term solar PV forecasting may not be the best for other applications involving heterogeneous data inputs. We hope that the architectural ideas for combining heterogeneous inputs, as described in the literature review and used in this study, will be useful to others focusing on other applications.

Setting up hundreds of experiments, tracking their progress, analyzing results, and conducting further experiments accordingly were extremely time-consuming. Was doing all this by ourselves, the best use of our time as researchers? Could we have been more efficient? What fraction of this pipeline could have been automated? Have the inefficiencies and/or our bias limited our search space and thus prevented us from finding the best architecture? These are valid questions that are being addressed by the machine learning research community with the advent of neural architecture search (NAS), the process of automating architecture engineering.[44] NAS is a subset of automated machine learning (AutoML),[45] which is an end-to-end approach that involves data cleaning and preprocessing, architecture engineering and assessment, and, in some cases, even deployment. Strategies explored in NAS include reinforcement learning,[46] Neuro-evolutionary approaches,[47] Bayesian optimization,[48] and direct gradient-based methods.[49] The process of NAS can still be time and resource intensive if a high number of candidate architectures are to be trained and validated. This is being addressed through ideas such as learning curve extrapolation[50] and searching for increasingly complex architectures, while simultaneously learning a surrogate model to guide the search through the search-space.[51] As noted in Ref. 44, most existing work in NAS has focused on the task of image-classification. Given the plethora of previous research in image classification, it is easy to define a search space for this task, which in some ways can still introduce some bias. Nevertheless, given the recent foray into other applications such as image restoration[52] and machine translation[53] and the demonstrated superiority of NAS derived models as compared to the manually designed ones, using NAS strategies for exploring architectures for heterogenous input fusion is certainly a path worth exploring.

ACKNOWLEDGMENTS

Experiments were conducted using the computational resources of the Stanford School of Earth's Center for Computational Earth and Environmental Science and the Google Cloud Platform (GCP). Google Cloud credits were obtained gratis via the GCP research credits program.[54]

Published under license by AIP Publishing

APPENDIX A: ARCHITECTURE DIAGRAMS FOR ALL 28 METHODS OF FUSION

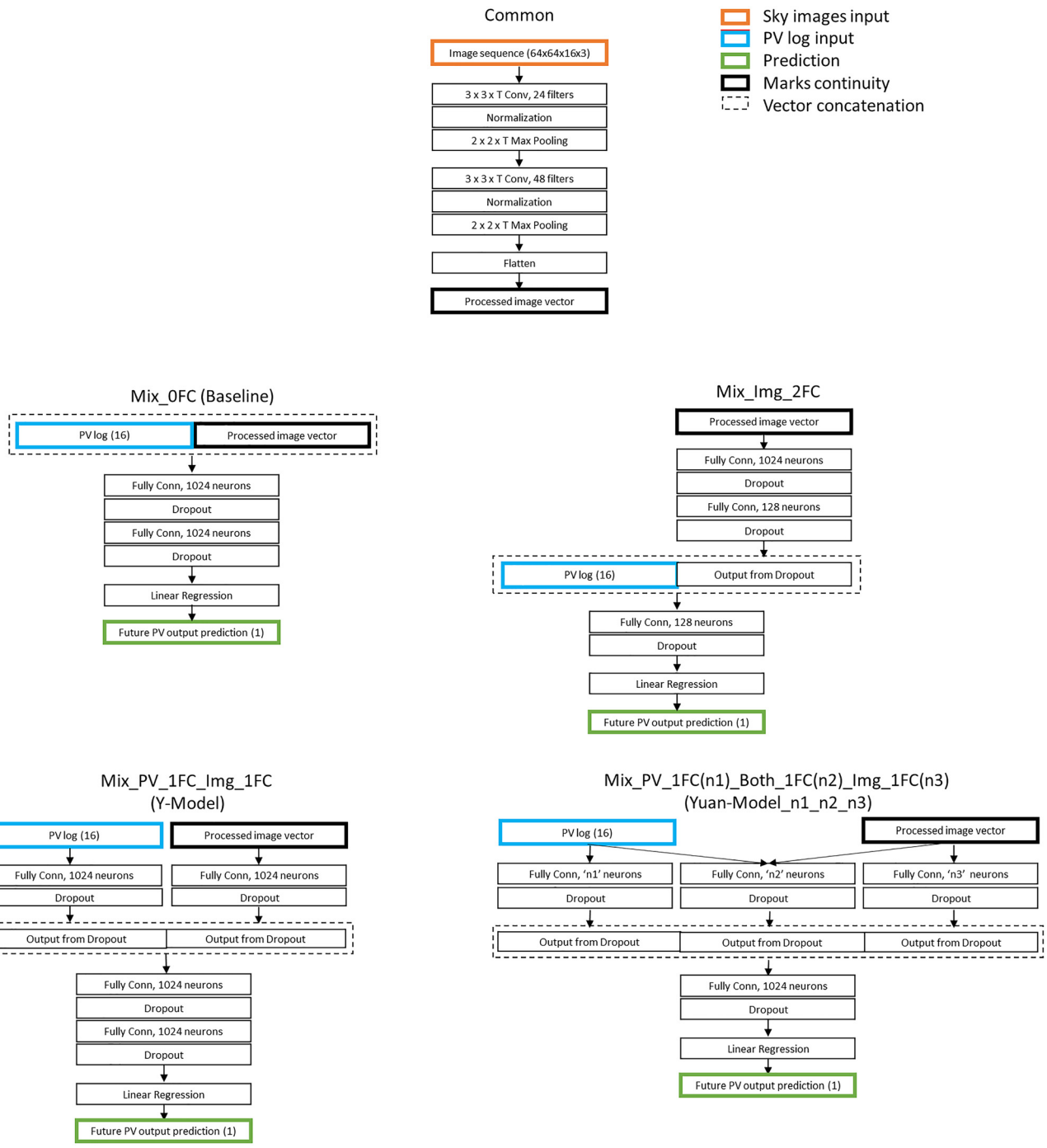

FIG. 15. Architectural details for the mix-in type of MoF. The first half of the architecture is common. It yields a processed image vector which is then fused with PV log in different ways and processed further yielding final prediction. The Y-model and the Yuan-model are named as such given their resemblance to the letter "Y" and the currency symbol for Yuan, "¥," respectively.

Published under license by AIP Publishing

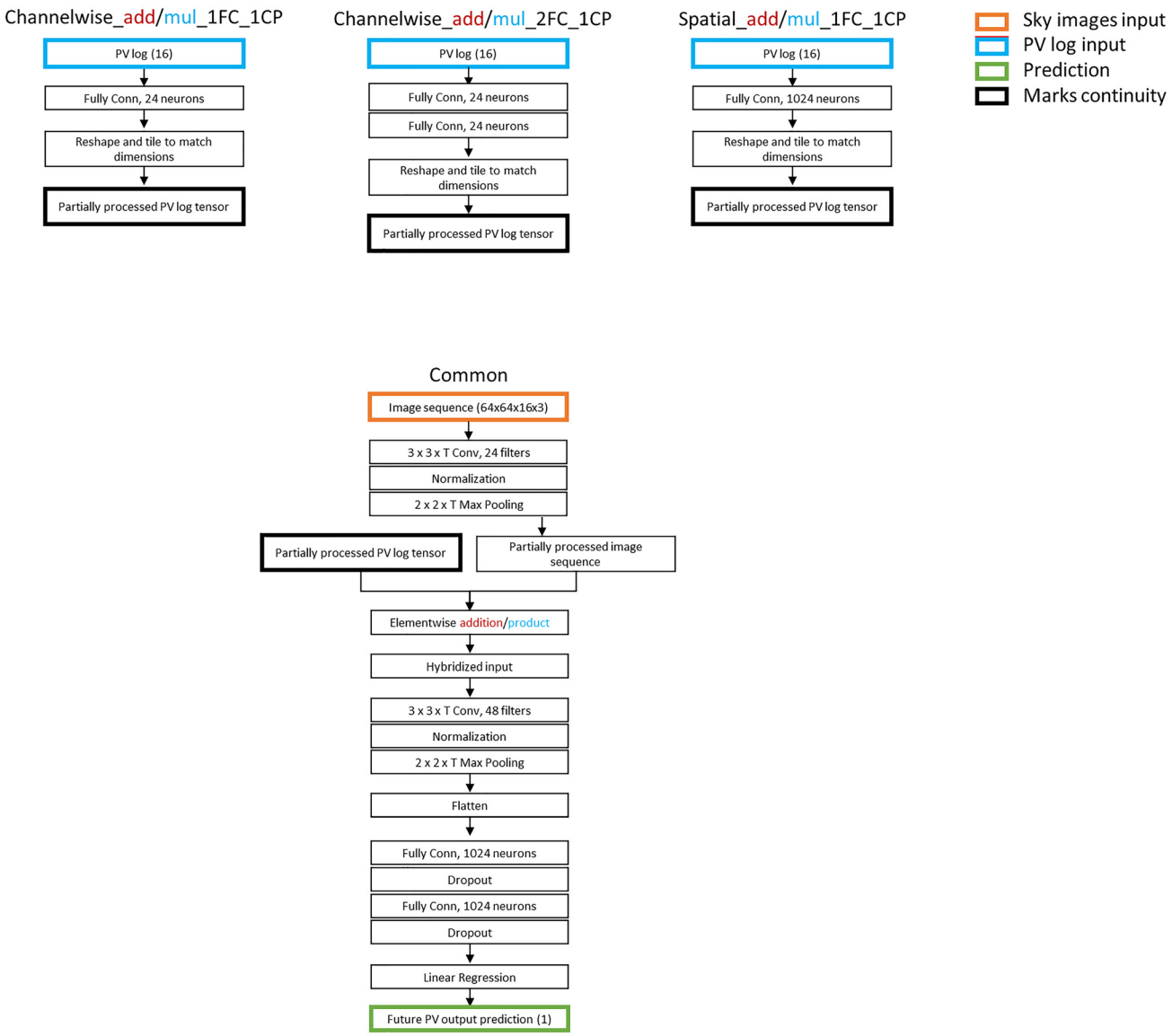

FIG. 16. Architectural details for the activation map combination, MoF. The first half of the architecture processes PV log in slightly different ways and then reshapes and tiles the resulting vector to match the dimensions of the partially processed image sequence. The second (common) half of the architecture fuses the two tensors and processes the resulting hybridized input further.

Published under license by AIP Publishing

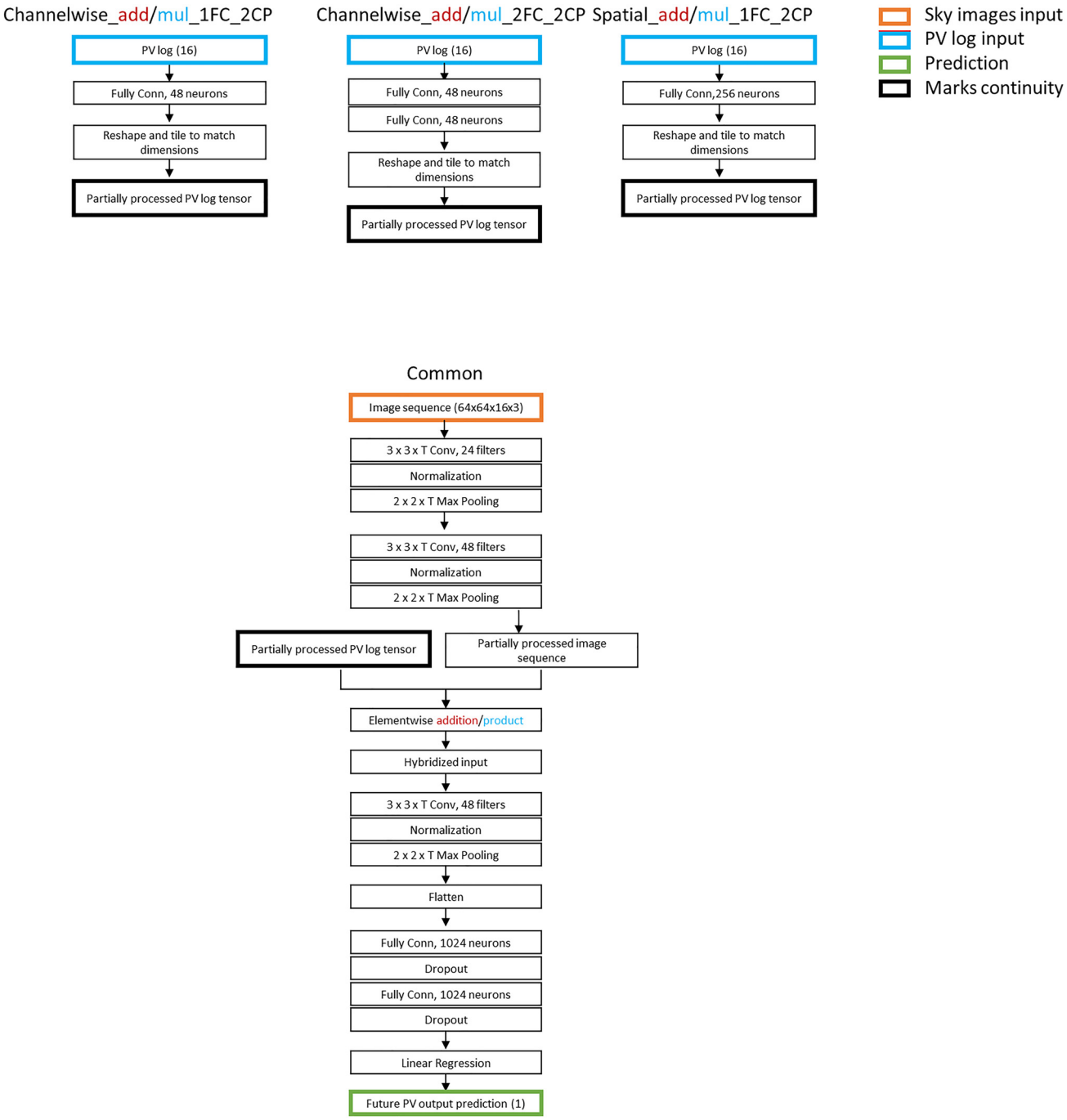

FIG. 17. Architectural details for the remaining activation map combination, MoF. The first half of the architecture processes PV log in slightly different ways and then reshapes and tiles the resulting vector to match the dimensions of the partially processed image sequence. The second (common) half of the architecture fuses the two tensors and processes the resulting hybridized input further.

Published under license by AIP Publishing

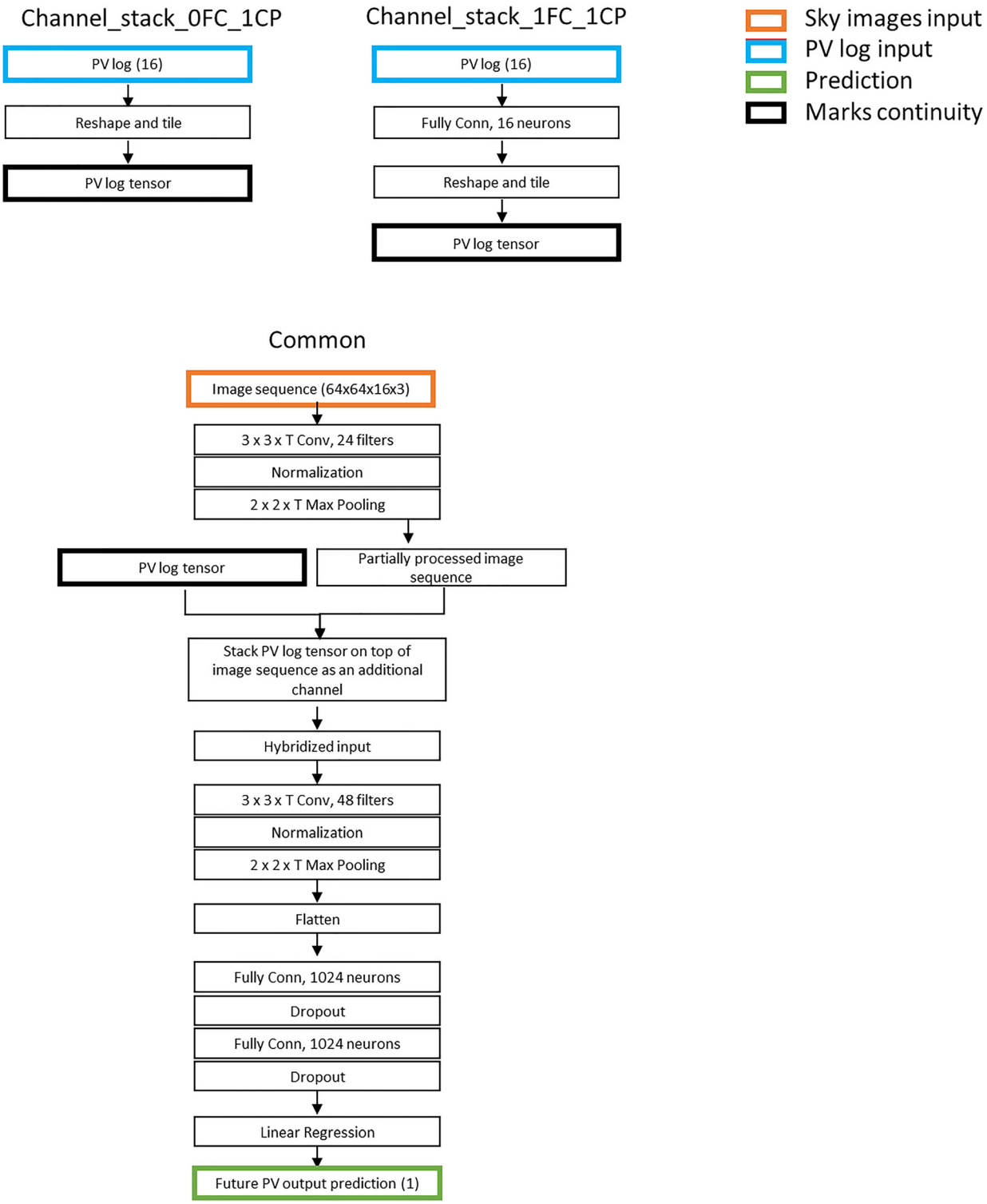

FIG. 18. Architectural details for the activation map stacking MoF. The first half of the architecture processes PV log in slightly different ways and then reshapes and tiles the resulting vector to match the dimensions of the partially processed image sequence. The second (common) half of the architecture stacks the PV log-based tensor as an additional channel onto the image-based tensor. The two tensors and processes the resulting hybridized input further.

Published under license by AIP Publishing

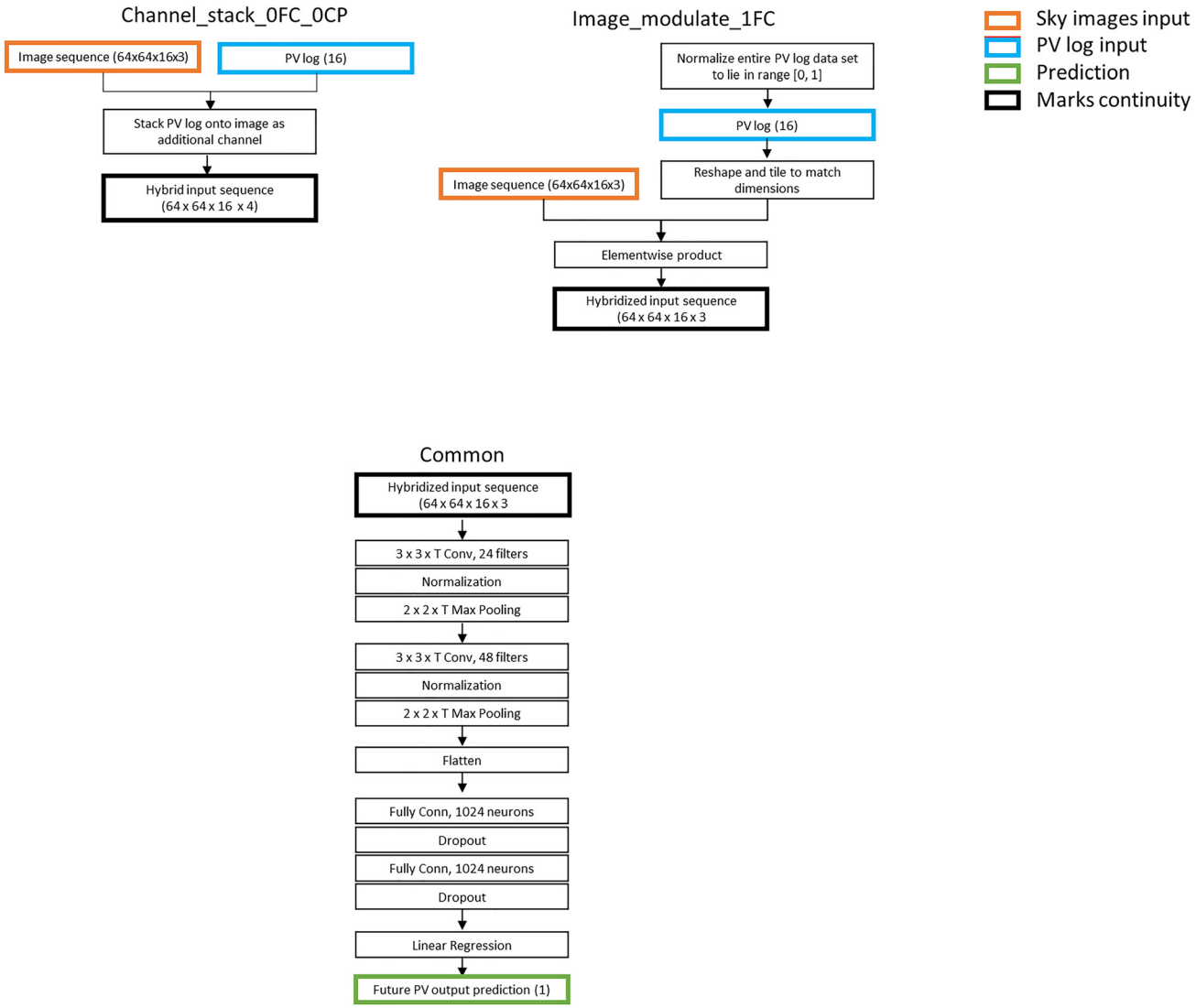

FIG. 19. Architectural details for the MoF wherein the image-based input is modified with the PV log input and then passed into a CNN.

Published under license by AIP Publishing

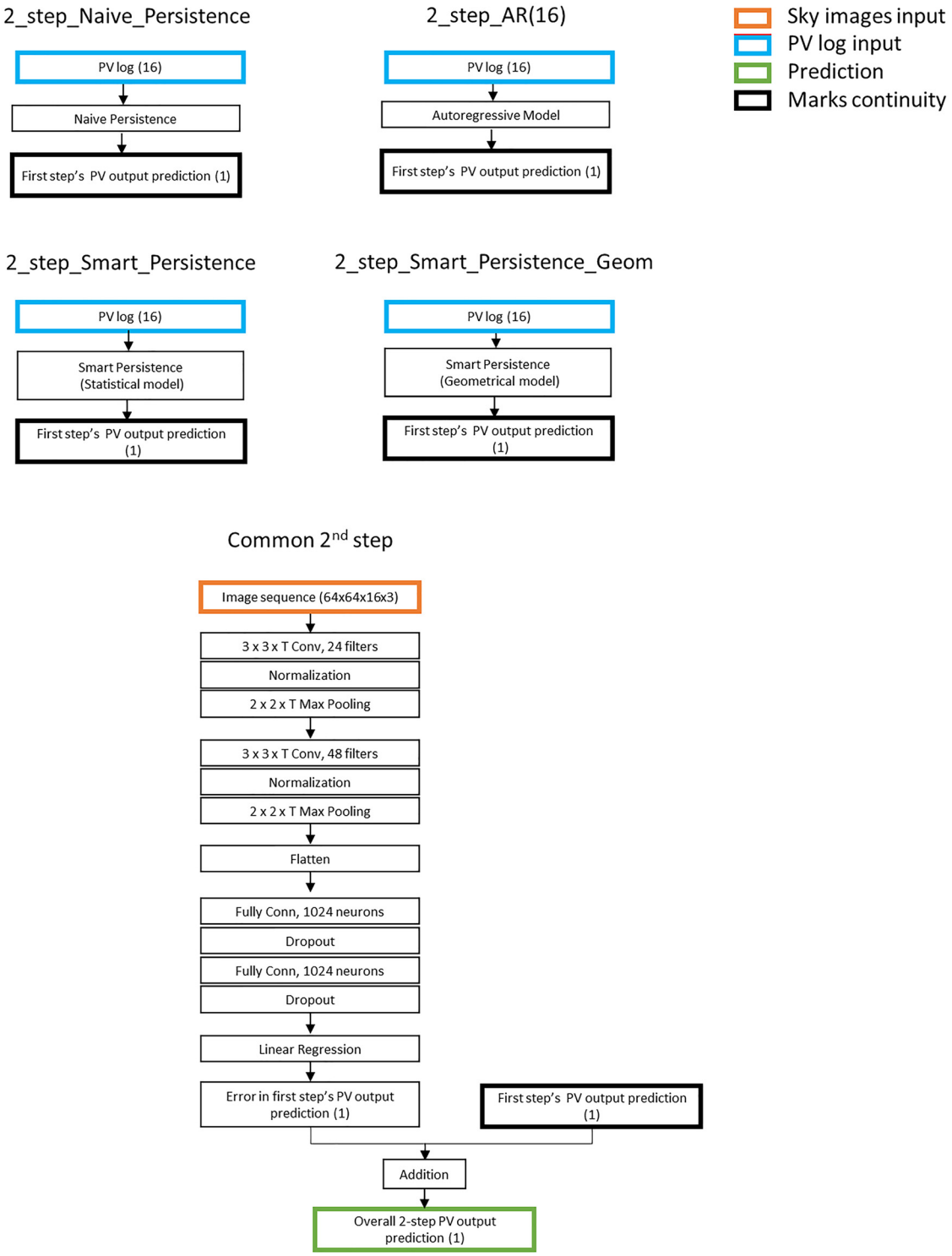

FIG. 20. Architectural details for the 2_step MoF. PV log is used to predict future output using a simple time-series model. A CNN model with the Baseline MoF then predicts the error in the prediction. Both predictions are combined to then yield the final prediction. Two approaches are used to generate the clear sky outputs used for smart persistence forecast. One is based on a geometric model,[55] while the other is based on a statistical clear sky fitting algorithm.[56]

Published under license by AIP Publishing

APPENDIX B: MOF AND HYPERPARAMETER CHOICES IN EACH OF THE FOUR STAGES OF EXPERIMENTATION

Stage 1

Method of Fusion	Baseline, Y-Model, Mix_Img_2FC, Channelwise_add_1FC_1CP, Channelwise_add_2FC_1CP, Channelwise_add_1FC_2CP, Channelwise_add_2FC_2CP, Spatial_add_1FC_1CP, Spatial_add_1FC_2CP, Channelwise_mul_1FC_1CP, Channelwise_mul_2FC_1CP, Channelwise_mul_1FC_2CP, Channelwise_mul_2FC_2CP, Spatial_mul_1FC_1CP, Spatial_mul_1FC_2CP, Image_modulate_1FC, Channel_stack_0FC_0CP, Yuan-Model_512_64_128, Yuan-Model_512_512_128, Yuan-Model_512_64_256, Yuan-Model_512_16_256, Yuan-Model_512_64_512, Channel_stack_0FC_1CP, Channel_stack_1FC_1CP, 2_step_AR(16), 2_step_Naive_Persistence, 2_step_Smart_Persistence, 2_step_Smart_Persistence_Geom
Learning rate	5e-7, 4e-6, 5e-6, 1.5e-5
Drop rate	0.1, 0.4, 0.5
Activation negative slope	0.01, 0.05, 0.1
Z-normalization beforehand	Yes, No
Normalization strategy within architecture	Batch, Layer, Instance, None
Convolution temporal depth	(1, 1), (1, 3), (3, 1), (3, 3), (3, 5), (5, 3), (5, 5), (5, 7), (7, 5), (7, 7), (2D, 2D)
Pooling temporal depth	(1, 1), (1, 2), (2, 2), (2D, 2D)

Stage 2

Method of Fusion	Y-Model, Spatial_add_1FC_1CP, Yuan-Model_128_64_512, Yuan-Model_128_512_512, Yuan-Model_256_64_512, Yuan-Model_256_16_512, Yuan-Model_512_64_512, 2_step_AR(16)
Learning rate	4e-6, 7e-6, 1.5e-5
Drop rate	0.1, 0.25, 0.4
Activation negative slope	0.01, 0.05, 0.1
Z-normalization beforehand	Yes, No
Normalization strategy within architecture	Batch, Layer, None
Convolution temporal depth	(1,1), (3,1),(3,3),(3,5),(5,3),(5,7),(2D, 2D)
Pooling temporal depth	(1, 1), (1, 2), (2, 2), (2D, 2D)

Stage 3

	Y-Model	Spatial_add_1FC_1CP	Yuan-Model_512_64_512	2_step_AR(16)
Learning rate	8e-6, 1.34e-5, 1.36e-5	3.6e-6, 4.1e-6, 6.5e-6	8e-6, 1.04e-5, 1.54e-5	4.5e-6, 7.2e-6, 1.27e-5
Drop rate	0.12, 0.22, 0.4	0.1, 0.23, 0.42	0.12, 0.27, 0.4	0.1, 0.27, 0.34
Activation negative slope	0.01, 0.056, 0.1	NA	0.068, 0.07, 0.09	NA
Z-normalization beforehand	Yes	No	Yes	No
Normalization strategy within architecture	Layer, None	Batch, Layer	None	Layer
Convolution temporal depth	(1,1), (5,3),(5,7),(2D,2D)	(3,3),(3,5),(5,7),(2D,2D)	(1,1), (3,3), (5,3), (5,7)	(2D,2D)
Pooling temporal depth	(1,1), (1,2), (2D,2D)	(1,1),(1,2),(2,2),(2D,2D)	(1,1), (1,2), (2,2)	(2D,2D)

Final training and testing on entirety of dataset

	Y-Model	Spatial_add_1FC_1CP	Yuan-Model_512_64_512	2_step_AR(16)
Method of Fusion	Y-Model	Spatial_add_1FC_1CP	Yuan-Model_512_64_512	2_step_AR(16)
Learning rate	1.34e-5	4.1e-6	1.04e-5	7.2e-6
Drop rate	0.22	0.23	0.27	0.27
Activation negative slope	0.056	NA	0.07	NA
Z-normalization beforehand	Yes	No	Yes	No
Normalization strategy within architecture	None	Batch	None	Layer
Convolution temporal depth	(5,3)	(3,3)	(3,3)	(2D,2D)
Pooling temporal depth	(1,1)	(1,2)	(1,2)	(2D,2D)

FIG. 21. Progression of the study through 4 stages of experimentation. The MoF and hyperparameters used in every stage are shown in two colors. Choices marked in black are retained for the next stage, while those marked in red are not.

Published under license by AIP Publishing

REFERENCES

[1]Y. Chu, B. Urquhart, S. M. I. Gohari, H. T. C. Pedro, J. Kleissl, and C. F. M. Coimbra, "Short-term reforecasting of power output from a 48 MWe solar PV plant," Sol. Energy **112**, 68–77 (2015).

[2]V. P. A. Lonij, A. E. Brooks, A. D. Cronin, M. Leuthold, and K. Koch, "Intra-hour forecasts of solar power production using measurements from a network of irradiance sensors," Sol. Energy **97**, 58–66 (2013).

[3]R. Dambreville, P. Blanc, J. Chanussot, and D. Boldo, "Very short term forecasting of the Global Horizontal Irradiance using a spatio-temporal autoregressive model," Renewable Energy **72**, 291–300 (2014).

[4]J. Zhang, R. Verschae, S. Nobuhara, and J.-F. Lalonde, "Deep photovoltaic nowcasting," Sol. Energy **176**, 267–276 (2018).

[5]M. Rana, I. Koprinska, and V. G. Agelidis, "Univariate and multivariate methods for very short-term solar photovoltaic power forecasting," Energy Convers. Manage. **121**, 380–390 (2016).

[6]G. Reikard, "Predicting solar radiation at high resolutions: A comparison of time series forecasts," Sol. Energy **83**(3), 342–349 (2009).

[7]A. Moreno-Munoz, J. J. G. de la Rosa, R. Posadillo, and F. Bellido, "Very short term forecasting of solar radiation," in 2008 33rd IEEE Photovoltaic Specialists Conference (2008), pp. 1–5.

[8]Y. Sun, G. Szucs, and A. R. Brandt, "Solar PV output prediction from video streams using convolutional neural networks," Energy Environ. Sci. **11**(7), 1811–1818 (2018).

[9]Y. Sun, V. Venugopal, and A. R. Brandt, "Short-term solar power forecast with deep learning: Exploring optimal input and output configuration," Sol. Energy **188**, 730–741 (2019).

[10]D. Xu, D. Anguelov, and A. Jain, "PointFusion: Deep sensor fusion for 3D bounding box estimation," in IEEE Conference on Computer Vision and Pattern Recognition (2018), pp. 244–253.

[11]E. Ayrey, D. J. Hayes, J. B. Kilbride, and A. R. Weiskittel, "Synthesizing disparate LiDAR and satellite datasets through deep learning to generate wall-to-wall forest inventories of New England," 580514 (2019).

[12]Y. Zhou and K. Hauser, "Incorporating side-channel information into convolutional neural networks for robotic tasks," in 2017 IEEE International Conference on Robotics and Automation (ICRA) (2017), pp. 2177–2183.

[13]Y. Zhou and K. Hauser, "6DOF grasp planning by optimizing a deep learning scoring function," in *Robotics: Science and Systems (RSS) Workshop on Revisiting Contact – Turning a Problem into a Solution* (Springer, 2017)

[14]I. Lenz, H. Lee, and A. Saxena, "Deep learning for detecting robotic grasps," Int. J. Rob. Res. **34**(4–5), 705–724 (2015).

[15]J. Ngiam and A. Y. Ng, "Multimodal deep learning," in Proceedings of the 28th International Conference on Machine Learning, Bellevue, WA, 2011.

[16]N. Srivastava, "Multimodal learning with deep Boltzmann machines," in Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (2013), pp. 329–336.

[17]D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," preprint arXiv:1412.6980 (2014).

[18]N. Shrivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," J. Mach. Learn. Res. **15**(1), 1929–1958 (2014).

[19]S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," Proceedings of the 32nd International Conference on Machine Learning, (PMLR, 2015), Vol. 37, pp. 448–456.

[20]J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," preprint arXiv:1607.06450.

[21]D. Ulyanov and A. Vedaldi, "Instance normalization: The missing ingredient for fast stylization," preprint arXiv:1607.08022.

[22]V. Nair and G. E. Hinton, "Rectified linear units improve restricted Boltzmann machines," in the Proceedings of the 27th International Conference on Machine Learning, Haifa, Israel, 2010).

[23]See http://cs231n.github.io/neural-networks-1/ for "CS-231N, Course Notes, Stanford University."

[24]D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri, "Learning spatiotemporal features with 3D convolutional networks," (IEEE, 2015). pp. 4489–4497.

[25]T. Hastie, R. Tibshirani, J. Gareth, and W. Daniela, *An Introduction to Statistical Learning with Applications in R* (Springer, 2017).

[26]J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization," J. Mach. Learn. Res. **13**, 281–305 (2012).

[27]F. A. C. Viana, "A tutorial on Latin hypercube design of experiments," Qual. Reliab. Eng. Int. (2016).

[28]M. Van der Merwe, Q. Lu, B. Sundaralingam, M. Matak, and T. Hermans, "Learning Continuous 3D Reconstructions for Geometrically Aware Grasping," eprint arXiv:1910.00983.

[29]W. Zhang and K. Hauser, "Single-image footstep prediction for versatile legged locomotion," in 2018 IEEE International Conference on Robotics and Automation (ICRA) (2018), pp. 4407–4413.

[30]Y. Zhu and T. Erez, "Reinforcement and imitation learning for diverse visuomotor skills," preprint arXiv:1802.09564.

[31]S. Levine, C. Finn, T. Darrell, and P. Abbeel, "End-to-end training of deep visuomotor policies," J. Mach. Learn. Res. **17**, 1–40 (2015).

[32]S. Koo, G. Ficht, G. M. García, D. Pavlichenko, M. Raak, and S. Behnke, "Robolink feeder: Reconfigurable bin-picking and feeding with a lightweight cable-driven manipulator," in 2017 13th IEEE Conference on Automation Science and Engineering (CASE) (2017), pp. 41–48.

[33]A. Yahya, A. Li, M. Kalakrishnan, Y. Chebotar, and S. Levine, "Collective robot reinforcement learning with distributed asynchronous guided policy search," in 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (2017), pp. 79–86.

[34]J. Mahler, J. Liang, S. Niyaz, M. Laskey, R. Doan, X. Liu, J. A. Ojea, and K. Goldberg, "Dex-net 2.0: Deep learning to plan robust grasps with synthetic point clouds and analytic grasp metrics," preprint arXiv:1703.09312.

[35]A. Dosovitskiy and V. Koltun, "Learning to act by predicting the future," International Conference on Learning Representations, 1–14 (2017).

[36]K. Fang, Y. Zhu, A. Garg, A. Kurenkov, V. Mehta, L. Fei-Fei, and S. Savarese, "Learning task-oriented grasping for tool manipulation from simulated self-supervision," arXiv:1806.09266.

[37]R. Calandra, A. Owens, D. Jayaraman, J. Lin, W. Yuan, J. Malik, E. Adelson, and S. Levine, "More than a feeling: Learning to grasp and regrasp using vision and touch," IEEE Robot. and Autom. Lett. **3**(4), 3300–3307 (2018).

[38]D. Quillen, E. Jang, O. Nachum, C. Finn, J. Ibarz, and S. Levine, "Deep reinforcement learning for vision-based robotic grasping: A simulated comparative evaluation of off-policy methods," in 2018 IEEE International Conference on Robotics and Automation (ICRA) (2018), pp. 6284–6291.

[39]K. Fang, Y. Bai, S. Hinterstoisser, S. Savarese, and M. Kalakrishnan, "Multi-task domain adaptation for deep learning of instance grasping from simulation," in 2018 IEEE International Conference on Robotics and Automation (ICRA) (2018), pp. 3516–3523.

[40]S. Levine, P. Pastor, A. Krizhevsky, J. Ibarz, and D. Quillen, "Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection," Int. J. Rob. Res. **37**(4-5), 421–436 (2018).

[41]M. Jaśkowski *et al.*, "Improved GQ-CNN: Deep learning model for planning robust grasps," arXiv:1802.05992.

[42]O. Vinyals, A. S. Vezhnevets, and D. Silver, "StarCraft II: A new challenge for reinforcement learning," arXiv:1708.04782.

[43]F. Sadeghi, A. Toshev, and E. Jang, "Sim2Real view invariant visual servoing by recurrent control," arXiv:1712.07642.

[44]T. Elsken, J. H. Metzen, and F. Hutter, "Neural architecture search: A survey," J. Mach. Learn. Res. **20**, 1–21(2019).

[45]F. Hutter, L. Kotthoff, and J. Vanschoren, *Automated Machine Learning: Methods, Systems, Challenges* (Springer, 2019).

[46]B. Zoph and Q. V. Le, "Neural architecture search with reinforcement learning," preprint arXiv:1611.01578 (2016).

[47]E. Real, A. Aggarwal, Y. Huang, and Q. V. Le, "Regularized evolution for image classifier architecture search," Proc. AAAI Conf. Artif. Intell. **33**, 4780–4789 (2019).

[48]H. Mendoza, A. Klein, M. Feurer, J. T. Springenberg, and F. Hutter, "Towards automatically-tuned neural networks," in Workshop on Automatic Machine Learning (2016), pp. 58–65.

[49]H. Liu, K. Simonyan, and Y. Yang, "DARTS: Differentiable architecture search," preprint arXiv:1806.09055 (2018).

[50]B. Baker, O. Gupta, R. Raskar, and N. Naik, "Accelerating neural architecture search using performance prediction," preprint arXiv:1705.10823 (2017).

Published under license by AIP Publishing

[51]C. Liu, B. Zoph, M. Neumann, J. Shlens, W. Huai, L. Jia, L. Fei-Fei, A. Yuille, J. Huang, and K. Murphy, "Progressive neural architecture search," in *European Conference on Computer Vision* (LNCS, 2018), Vol. 11205, pp. 19–35.
[52]M. Suganuma, M. Ozay, and T. Okatani, "Exploiting the potential of standard convolutional autoencoders for image restoration by evolutionary search," in 35th International Conference on Machine Learning, ICML (2018), Vol. 11, pp. 7592–7601.
[53]D. R. So, C. Liang, and Q. V. Le, "The evolved transformer," arXiv:1901.11117.
[54]See https://edu.google.com/programs/credits/faqs/?modal_active=none#research-credits for "Google Cloud Platform Research Credits Program."
[55]A. V. Da Rosa, *Fundamentals of Renewable Energy Processes* (Academic Press, 2012).
[56]E. Meyers, B. Tabone, and M. Kara, "Statistical clear sky fitting algorithm," in 45th IEEE Photovoltaic Specialist Conference (PVSC) (2018), pp. 1–6.

Published under license by AIP Publishing