Supplement of Nat. Hazards Earth Syst. Sci., 23, 3065–3077, 2023
https://doi.org/10.5194/nhess-23-3065-2023-supplement

Supplement of

Shallow and deep learning of extreme rainfall events from convective atmospheres

Gerd Bürger and Maik Heistermann

Correspondence to: Gerd Bürger (gbuerger@uni-potsdam.de)

Supplemental information

We generally assume the multivariate predictor vector is $\boldsymbol{x}$ and the binary predictand is $\boldsymbol{y}$.

The Shallow models

Lasso regression (LASSO)

In Lasso regression, the log-likelihood function of $\boldsymbol{y}$ given $\boldsymbol{\beta}$, $\log \ell(\boldsymbol{y}; \boldsymbol{\beta})$, is maximized with an additional penalty term λ for the size of the parameters:

$$\sum \log \ell(\boldsymbol{y}; \boldsymbol{\beta}) - \lambda |\beta|_1 \tag{S1}$$

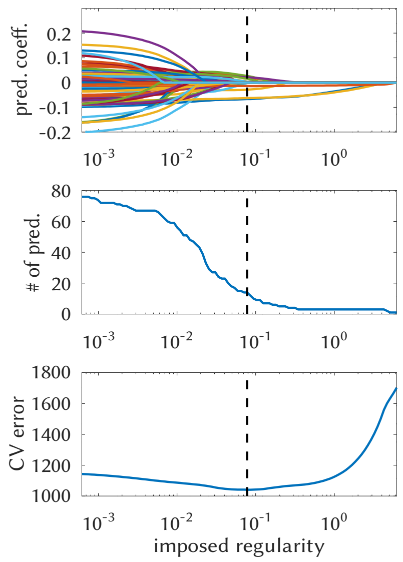

Figure S1. Result of Lasso regression. The panels show the dependency of three regression indicators on the imposed regularity (often denoted λ), the active predictors (top), the number of predictors (middle), and the corresponding cross-validation error.

With $\lambda \longrightarrow 0$ one obtains the classical maximum-likelihood estimate, and a growing λ squeezes the coefficients $\boldsymbol{\beta}$ towards zero; the optimum λ is found using cross-validation (but limited to the calibration set). We use the *penalized* toolbox (McIlhagga, 2016). A typical result is shown in Figure S1. It shows the successive disappearance of predictors with increasing regularity and how the cross-validated model error changes with that; its minimum is achieved with 14 predictors.

Random Forest (TREE)

Random forests are obtained from *bootstrapped averaging* ('bagging') of ensembles of decision trees, where the individual trees are obtained from resampling the data; each decision tree is grown following the 'M5 method'. For details and the corresponding software see the M5PrimeLab package, cf. http://www.cs.rtu.lv/jekabsons/regression.html. Here we use the M5PrimeLab toolbox of Gints Jēkabsons, cf. http://www.cs.rtu.lv/jekabsons/regression.html, with 200 trees to build (instead of the default 100). A typical tree for a case with 81 principal components as predictors is shown in Figure S2. To convert the final class prediction into a probabilistic one we simply calculate frequencies from the class predictions of the single trees.

Neural net (NNET)

This is a simple feed-forward backpropagation net with merely 2 layers, one with 7 and one with 3 nodes (neurons); the numbers are based on very basic testing, cf. https://octave.sourceforge.io/nnet. The result is stochastic, so we employ an ensemble of 20 realizations and calculate the class frequencies from the single predictions.

Logistic regression (NLS)

To generally map $\boldsymbol{x}$ to a probability we combine a normal linear mapping, given by a parameter vector $\boldsymbol{\beta}$, with the sigmoid function

$$\boldsymbol{x} \mapsto \frac{1}{1+e^{-\boldsymbol{\beta x}}} \tag{S2}$$

Following ordinary regression, the following expression needs to be minimized:

$$\sum \left(\frac{1}{1+e^{-\boldsymbol{\beta x}}} - \boldsymbol{y} \right)^2 \tag{S3}$$

for which a classical optimization procedure can be applied (here: Levenberg-Marquardt).

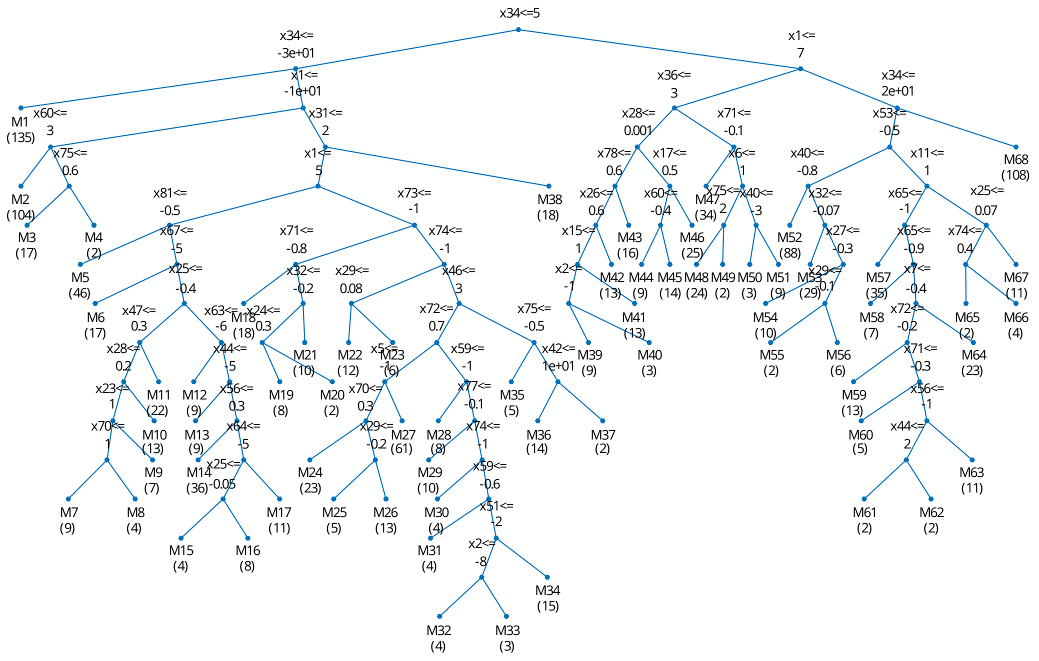

Figure S2. One sample model tree with 81 predictors (principal components), taken from an ensemble of 200 random trees.

Training CNNs with Caffe

Caffe is a framework for the training of CNNs, developed at the Berkeley AI Research (BAIR) and Berkeley Vision and Learning Center (BVLC), cf. https://github.com/BVLC/caffe. Because of its Matlab/Octave interface it was easily implemented in our programming framework at https://gitlab.dkrz.de/b324017/carlofff; all that was needed was two text files, a solver and a network file, containing a few crucial parameters that had to be adapted to the new context. The main body of their content could be left intact; for example, we generally use the stochastic gradient decent solver *Adam* and the *softmax* function for converting to probabilities, similar to the NLS approach above. Table S1 lists the hyperparameters (cf. https://github.com/BVLC/caffe/blob/master/src/caffe/proto/caffe.proto#L164-L165). All default settings can be inspected under https://gitlab.dkrz.de/b324017/carlofff/-/tree/master/models, with the model specific settings in the respective directory. No extra data split was done for the hyperparameter tuning, so the validation data are not fully independent. However, because the sole tuning criterion was not model skill but learning convergence, this should be negligible.

DWD warning level 3 vs. 5-year return period

Given that of the total of 175200 = 20×365×24 hours from 2001 to 2020, about 27000 are listed as extreme, the likelihood of seeing any extreme event in Germany is $p_G = 27000/175200 = 15\%$. The average size (in pixels) of a CatRaRE event is $a=133$, while all of Germany covers a_G=900×1100 = 990000 pixels. If all CatRaRE events can be taken as independent, then the probability of an event per pixel is $p = 1-(1-p_G)^{a_G/a} = 2.25\times10^{-5}$.

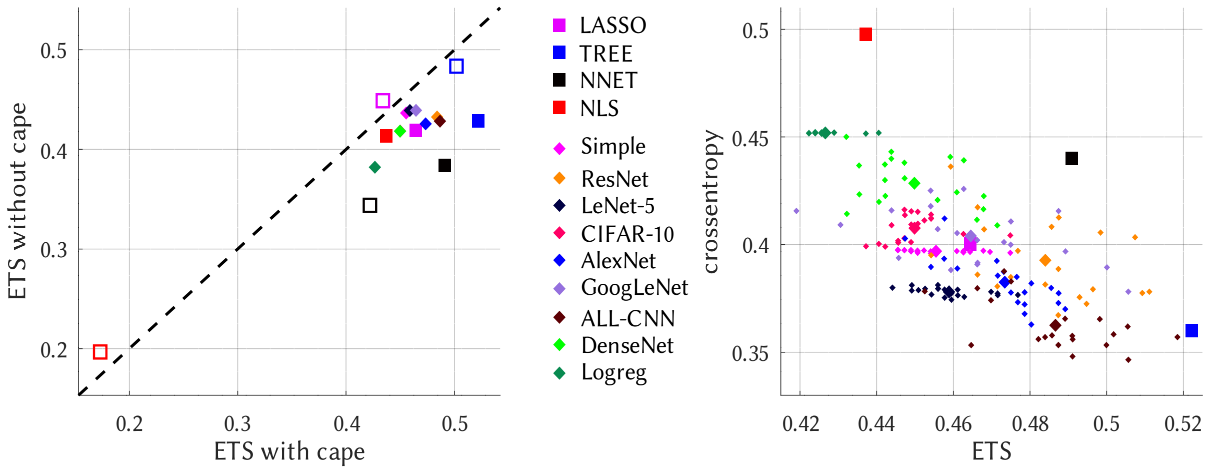

Figure S3. Like Fig. 3 using different realizations.

Table S1. Tuned hyperparameters for the networks, as modified from the original sources.

	max_iter	power	batch_size	base_lr
LeNet-5	1000	3	100	0.001
AlexNet	500	3	50	0.001
CIFAR-10	1000	3	10	0.001
ALL-CNN	1000	0.5	100	0.0017
GoogLeNet	500	3	10	0.001
ResNet	300	3	100	0.001
DenseNet	500	3	10	0.001
Simple	1000	3	100	0.001
Logreg	1000	3	100	0.001

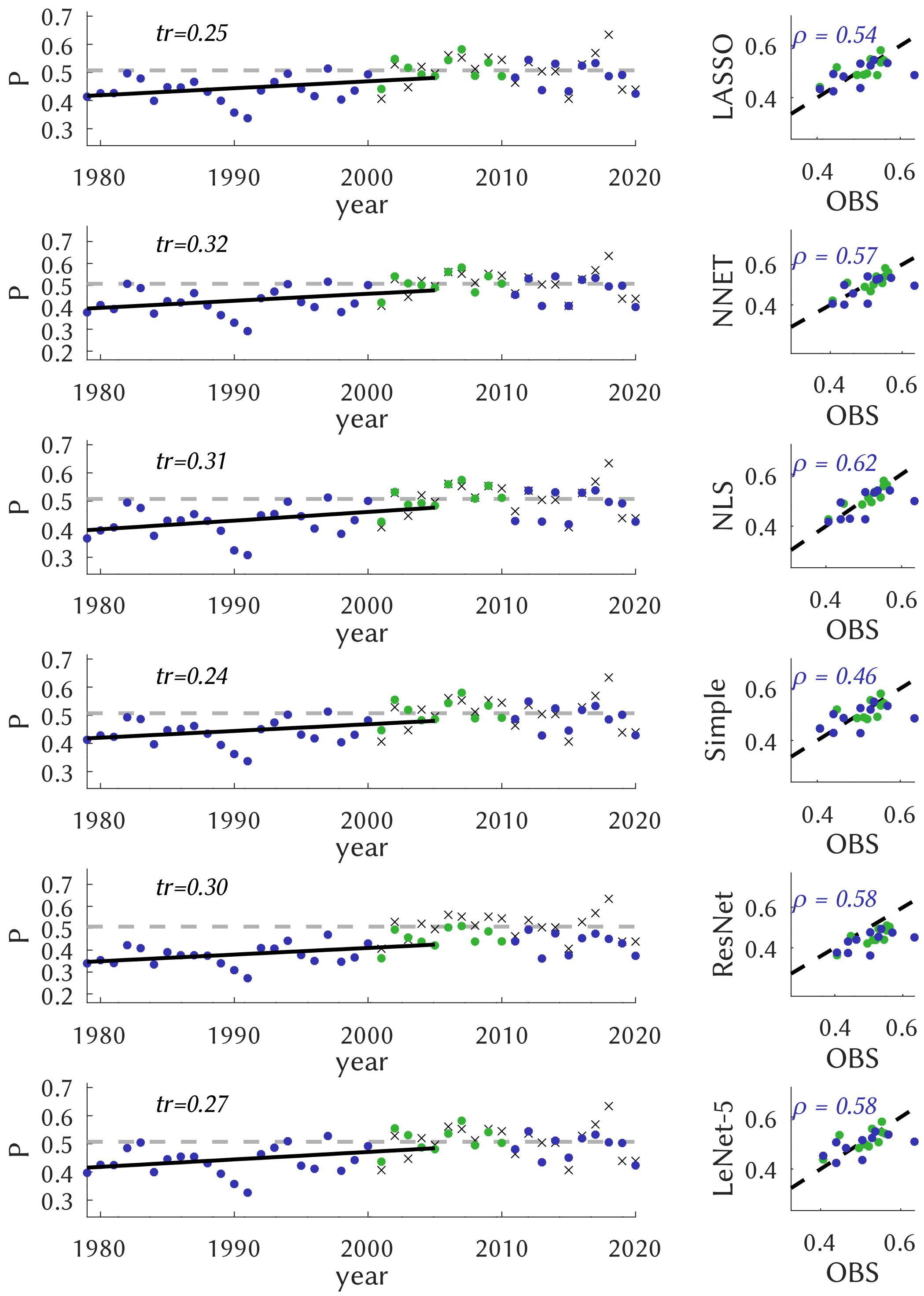

Figure S4. Similar to Fig. 6, for the remaining models (first half).

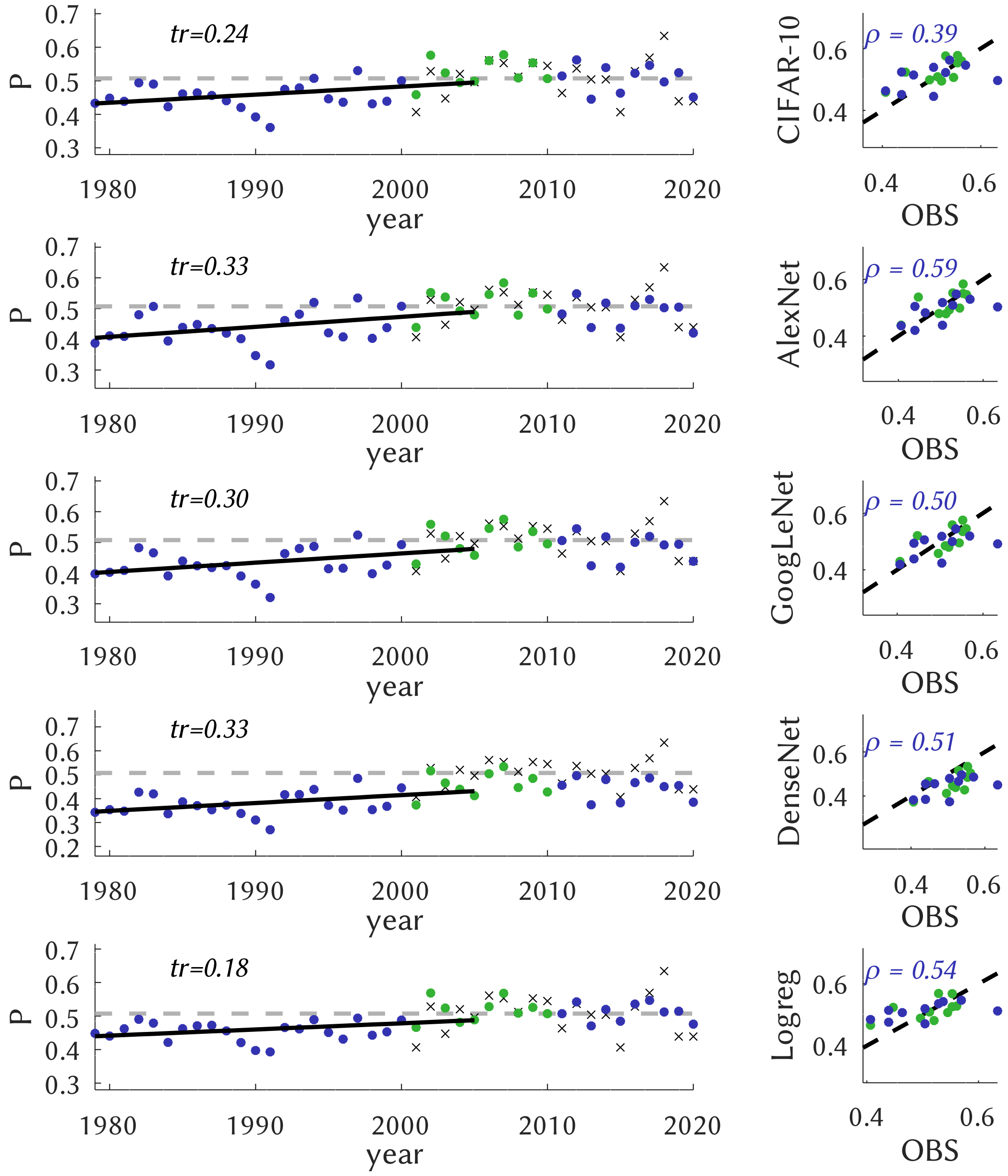

Figure S5. Fig. S4 continued.

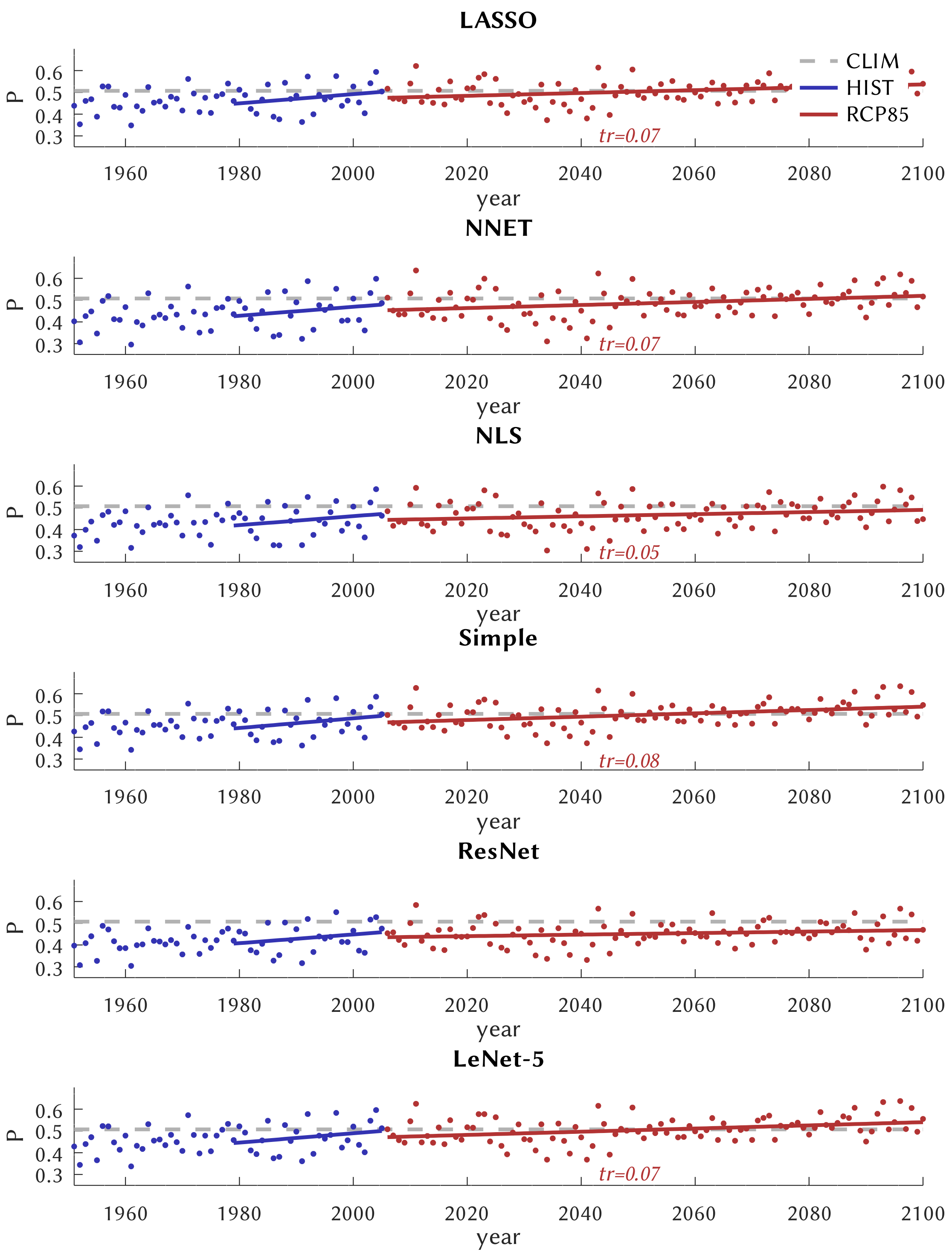

Figure S6. Like Figure 7, for the remaining models.

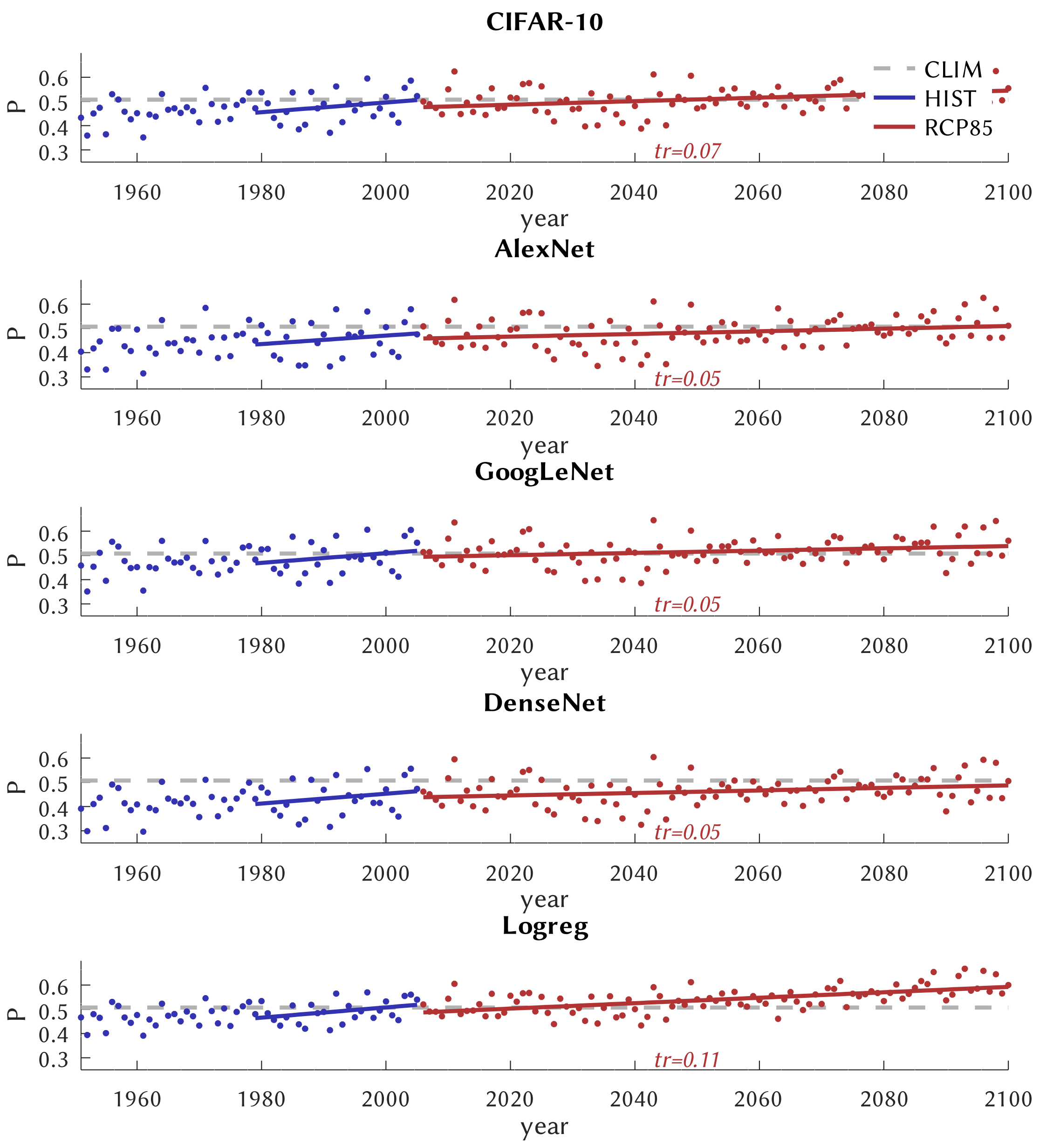

Figure S7. Fig. S6 continued.