Supplementary Materials

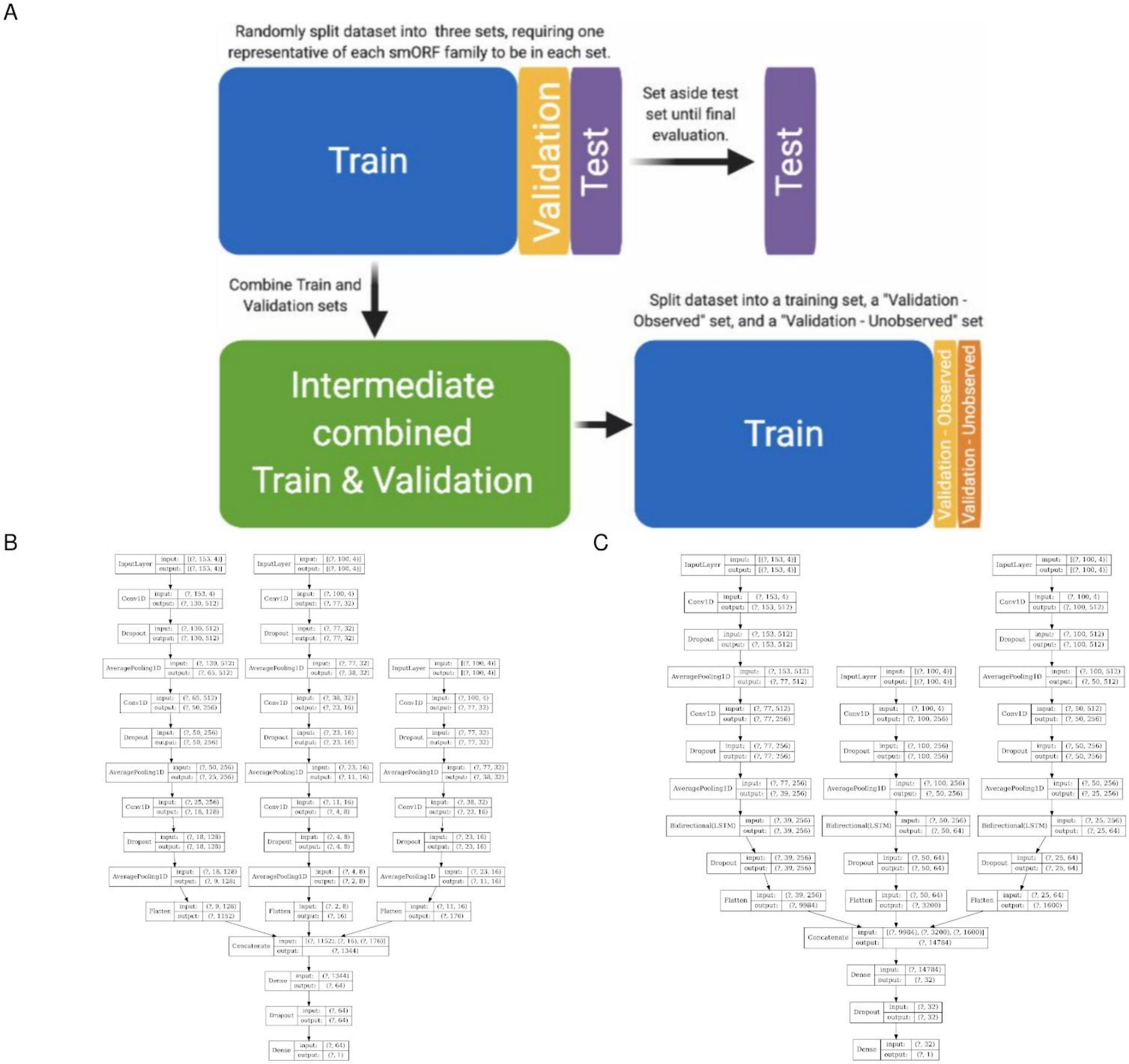

A
Randomly split dataset into three sets, requiring one representative of each smORF family to be in each set.
Train
Validation
Test
Set aside test set until final evaluation.
Test
Combine Train and Validation sets
Intermediate combined Train & Validation
Split dataset into a training set, a "Validation - Observed" set, and a "Validation - Unobserved" set
Train
Validation - Observed
Validation - Unobserved
B
InputLayer input: [(?, 153, 4)] output: [(?, 153, 4)]
Conv1D input: (?, 153, 4) output: (?, 130, 512)
Dropout input: (?, 130, 512) output: (?, 130, 512)
AveragePooling1D input: (?, 130, 512) output: (?, 65, 512)
Conv1D input: (?, 65, 512) output: (?, 50, 256)
Dropout input: (?, 50, 256) output: (?, 50, 256)
AveragePooling1D input: (?, 50, 256) output: (?, 25, 256)
Conv1D input: (?, 25, 256) output: (?, 18, 128)
Dropout input: (?, 18, 128) output: (?, 18, 128)
AveragePooling1D input: (?, 18, 128) output: (?, 9, 128)
Flatten input: (?, 9, 128) output: (?, 1152)
InputLayer input: [(?, 100, 4)] output: [(?, 100, 4)]
Conv1D input: (?, 100, 4) output: (?, 77, 32)
Dropout input: (?, 77, 32) output: (?, 77, 32)
AveragePooling1D input: (?, 77, 32) output: (?, 38, 32)
Conv1D input: (?, 38, 32) output: (?, 23, 16)
Dropout input: (?, 23, 16) output: (?, 23, 16)
AveragePooling1D input: (?, 23, 16) output: (?, 11, 16)
Conv1D input: (?, 11, 16) output: (?, 4, 8)
Dropout input: (?, 4, 8) output: (?, 4, 8)
AveragePooling1D input: (?, 4, 8) output: (?, 2, 8)
Flatten input: (?, 2, 8) output: (?, 16)
InputLayer input: [(?, 100, 4)] output: [(?, 100, 4)]
Conv1D input: (?, 100, 4) output: (?, 77, 32)
Dropout input: (?, 77, 32) output: (?, 77, 32)
AveragePooling1D input: (?, 77, 32) output: (?, 38, 32)
Conv1D input: (?, 38, 32) output: (?, 23, 16)
Dropout input: (?, 23, 16) output: (?, 23, 16)
AveragePooling1D input: (?, 23, 16) output: (?, 11, 16)
Flatten input: (?, 11, 16) output: (?, 176)
Concatenate input: [(?, 1152), (?, 16), (?, 176)] output: (?, 1344)
Dense input: (?, 1344) output: (?, 64)
Dropout input: (?, 64) output: (?, 64)
Dense input: (?, 64) output: (?, 1)
C
InputLayer input: [(?, 153, 4)] output: [(?, 153, 4)]
Conv1D input: (?, 153, 4) output: (?, 153, 512)
Dropout input: (?, 153, 512) output: (?, 153, 512)
AveragePooling1D input: (?, 153, 512) output: (?, 77, 512)
Conv1D input: (?, 77, 512) output: (?, 77, 256)
Dropout input: (?, 77, 256) output: (?, 77, 256)
AveragePooling1D input: (?, 77, 256) output: (?, 39, 256)
Bidirectional(LSTM) input: (?, 39, 256) output: (?, 39, 256)
Dropout input: (?, 39, 256) output: (?, 39, 256)
Flatten input: (?, 39, 256) output: (?, 9984)
InputLayer input: [(?, 100, 4)] output: [(?, 100, 4)]
Conv1D input: (?, 100, 4) output: (?, 100, 256)
Dropout input: (?, 100, 256) output: (?, 100, 256)
AveragePooling1D input: (?, 100, 256) output: (?, 50, 256)
Bidirectional(LSTM) input: (?, 50, 256) output: (?, 50, 64)
Dropout input: (?, 50, 64) output: (?, 50, 64)
Flatten input: (?, 50, 64) output: (?, 3200)
InputLayer input: [(?, 100, 4)] output: [(?, 100, 4)]
Conv1D input: (?, 100, 4) output: (?, 100, 512)
Dropout input: (?, 100, 512) output: (?, 100, 512)
AveragePooling1D input: (?, 100, 512) output: (?, 50, 512)
Conv1D input: (?, 50, 512) output: (?, 50, 256)
Dropout input: (?, 50, 256) output: (?, 50, 256)
AveragePooling1D input: (?, 50, 256) output: (?, 25, 256)
Bidirectional(LSTM) input: (?, 25, 256) output: (?, 25, 64)
Dropout input: (?, 25, 64) output: (?, 25, 64)
Flatten input: (?, 25, 64) output: (?, 1600)
Concatenate input: [(?, 9984), (?, 3200), (?, 1600)] output: (?, 14784)
Dense input: (?, 14784) output: (?, 32)
Dropout input: (?, 32) output: (?, 32)
Dense input: (?, 32) output: (?, 1)

Figure S1: Additional details regarding deep learning model training procedure and architecture, Related to Figure 1. (A) This schematic illustrates how the full dataset was split into training, validation, and test sets. First, the full dataset was split into training, validation, and test. Each division contains at least one representative of each smORF family. The rest of the examples were randomly assigned to each division, with approximately 80% being used as training, 10% for validation, and 10% for test. The test set was set aside and was not used for any of model architecture selection and hyperparameter tuning. The training and validation sets were recombined, and reallocated in such a way that the validation set was approximately 50% unobserved smORF families (not found in the training set) and 50% observed smORF families. Model architecture selection and hyperparameter tuning was done on this dataset, and performance in both the “Validation - Observed” and “Validation - Unobserved” datasets were considered when selecting the final DSN1 and DSN2 models, respectively. (B and C) Schematics representing the model architectures of the

DSN1 (B) and DSN2 (C) deep learning models. These were the final models chosen after an extensive hyperparameter tuning process. Each candidate smORF is broken up into three parts - the smORF sequence itself (leftmost branch in each model), 100 nucleotides upstream of the smORF (the middle branch), and 100 nucleotides downstream of the smORF. The input and output dimensions of each matrix in each layer is given. (B) The DSN1 model includes three convolutional layers to process the smORF sequence, with 512, 256, and 128 filters per layer of sizes 24, 16, and 8, respectively. The upstream sequence is processed by three convolutional layers, with 32, 16, and 8 filters per layer of sizes 24, 16, and 8, respectively. The downstream sequence is processed by two convolutional layers, with 32 and 16 filters per layer of sizes 24 and 16, respectively. The dropout rate used to regularize the weights of convolutional layers was 0.5, "valid" padding was used with each layer, and average pooling was used after each convolutional layer. The final dense layer that analyzes the flattened and concatenated output of all three branches includes 64 neurons, and a dropout rate of 0.5. Adam optimization (a commonly used optimization technique in deep learning that combines RMSprop and SGD with momentum) with a learning rate of 1e-4 was used to train the model (Kingma & Ba, 2014). (C) The DSN2 model includes two convolutional layers to process the smORF sequence, with 512 and 256 filters per layer of sizes 6 and 4, respectively, and a long short-term memory (LSTM) layer containing 128 neurons. The upstream sequence is processed by one convolutional layer with 256 filters of length 12, and an LSTM layer containing 32 neurons. The downstream sequence is processed by two convolutional layers with 512 and 256 filters per layer of sizes 24 and 16, respectively, and an LSTM layer containing 32 neurons. The dropout rate used to regularize the weights of convolutional layers was 0.5, the dropout rate for the LSTM layers was 0.3, "same" padding was used with each layer, and average pooling was used after each convolutional layer. The final dense layer that analyzes the flattened and concatenated output of all three branches includes 32 neurons, and a dropout rate of 0.5. Adam optimization with a learning rate of 1e-3 was used to train the model.

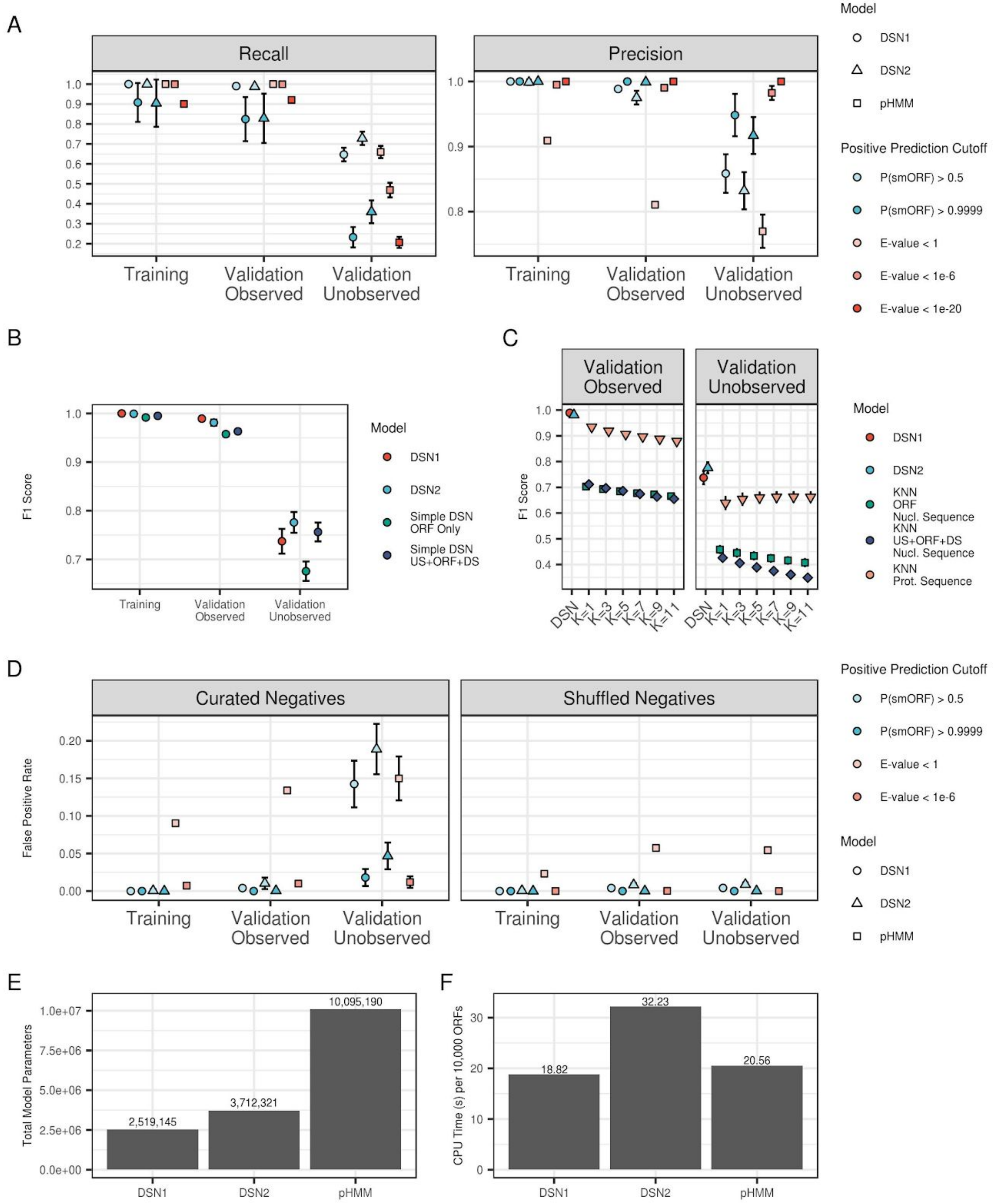


Figure S2: Performance characteristics of DSN smORF identification models, Related to Figure 1. (A) Each point is the average value after running the training procedure 64 times with randomly selected families excluded from the training set and assigned to the “Validation - Unobserved Families” set. Showing precision and recall for the three sets with different significant cutoffs. Error bars are the standard error of the 64 randomized samples. (B) The average F1 scores of simple neural networks compared to DSN1 and DSN2

across training and validation sets. “Simple DSN ORF only” refers to a neural network with a single layer, 144,001 total parameters, and a single input analyzing only the ORF nucleotide sequence. “Simple DSN ORF only US+ORF+DS” refers to a neural network with a single layer, 232,641 total parameters, and all three inputs analyzing the ORF, the upstream, and the downstream nucleotide sequences, respectively. (C) A comparison between the DSN models and k-nearest neighbors classifiers built to analyze the ORF nucleotide k-mer composition (KNN ORF Nucl. Sequence), the combination of the upstream, ORF, and downstream k-mer composition (KNN US+ORF+DS Nucl. Sequence), and the protein sequence kmer composition (KNN Prot. Sequence). Along the x-axis, K=1 to K=11 refers to the number of nearest neighbors that are considered when classifying each example. (D) The false positive rate (FPR) of the DSN and pHMM models across significance cutoffs and negative example categories. “Curated Negatives” are those negative examples that are naturally occurring but were considered negatives due to a lack of conservation signal (see Methods). “Shuffled Negatives” are synthesized negatives that were generated by tetramer shuffling of true positives and curated negatives. (E) Comparison of the total parameters in the finalized DSN1, DSN2, and pHMM models. (F) Comparison of the CPU time to process 10,000 ORFs across DSN1, DSN2, and pHMM models.

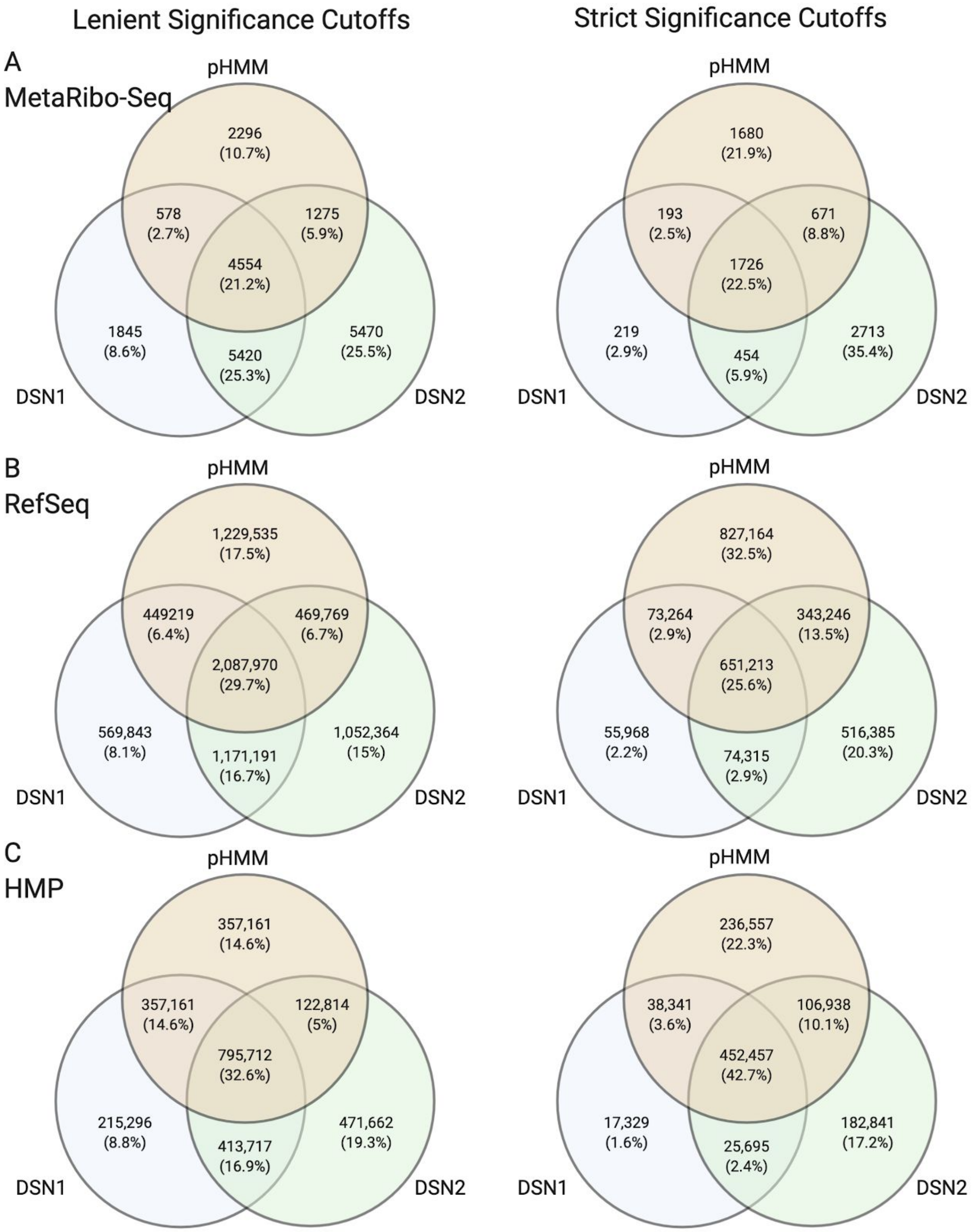


Figure S3: Overlap of smORF predictions across MetaRibo-Seq, RefSeq, and HMP datasets, Related to Figure 2. Venn diagrams indicating the overlap in the positive predictions in the MetaRibo-Seq (A), RefSeq (B) and HMP (C) datasets. The left column indicates the overlaps when using lenient significance cutoffs (pHMM E-value < 1, DSN1 P(smORF) > 0.5, and DSN2 P(smORF) > 0.5), and the right column indicates the overlaps when using the strict significance cutoffs (pHMM E-value < 1e-6, DSN1 P(smORF) > 0.9999, and DSN2 P(smORF) > 0.9999). The total number of positive predictions and the percentage of all positive predictions is given in each cell of the diagram.

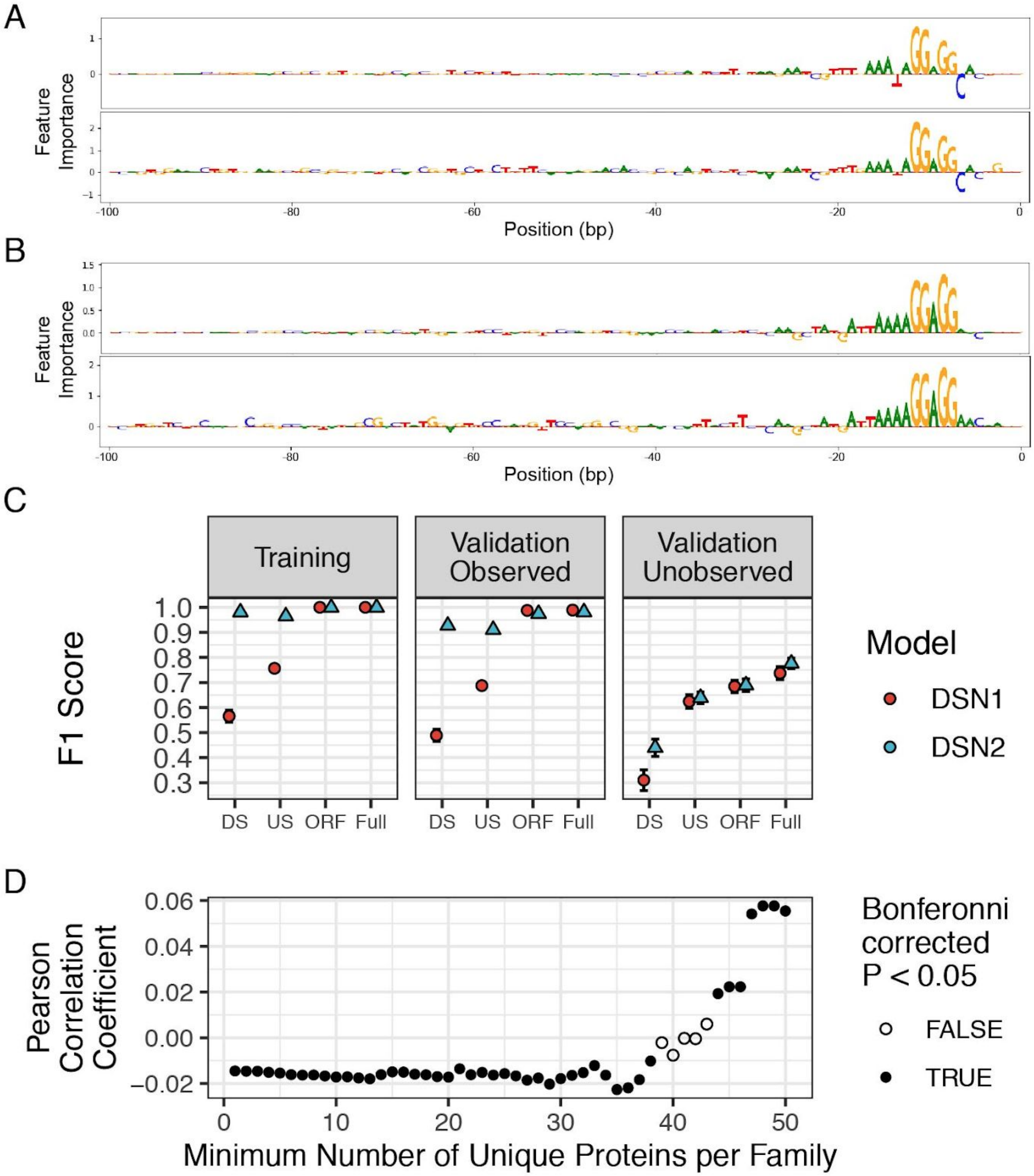
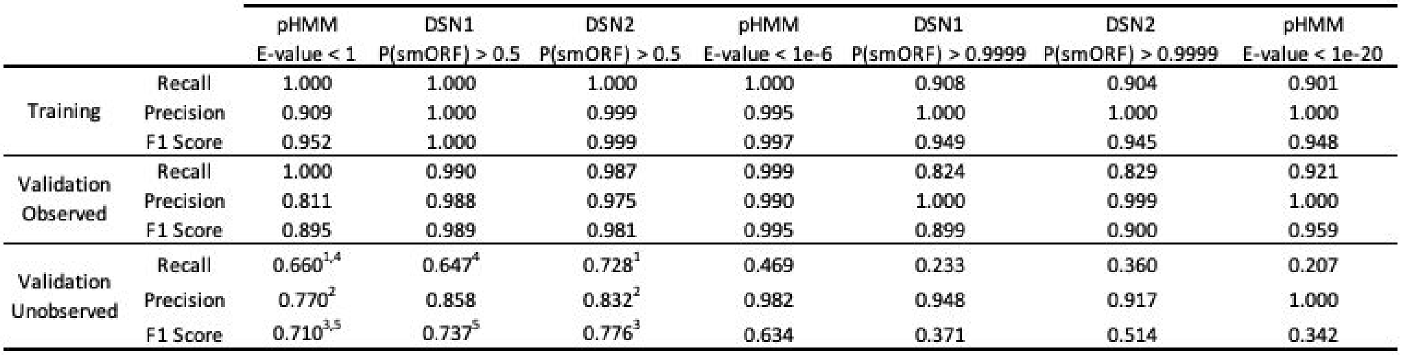


Figure S4: Additional details of feature importance analysis and feature ablation experiments, Related to Figure 3. (A and B) Showing the feature importance scores of upstream sequences as calculated using the DeepLIFT algorithm for two specific examples. This was implemented in the SHAP python package. The canonical AGGAGG Shine-Dalgarno motif is recognizable, with high importance being assigned to the two guanine repeats. (A) Feature importance scores of the upstream sequence of a single smorfam02316 example as assigned by DSN1 (top) and DSN2 (bottom). (B) Feature importance scores of the upstream sequence of a single smorfam03028 example as assigned by DSN1 (top) and DSN2 (bottom). (C) The average F1 score of modified DSN1 and DSN2 models that were retrained on the 64 randomized

datasets. Includes models that only consider the downstream nucleotide sequence (DS), the upstream nucleotide sequence (US), and the ORF sequence alone. The full model refers to the final, optimized DSN models presented in the main text. All hyperparameters outside of the deleted input branches were kept identical across the modified models. (D) The correlation between the feature importance score as calculated by DeepLIFT and positional entropy as calculated from microprotein family multiple sequence alignments. Black points indicate a statistically significant correlation, white points are not significant. This correlation is re-calculated at different cutoffs along the x-axis, where the cutoff indicates the minimum number of unique microproteins per family.

Term	Description
Recall	A metric to measure the performance of a predictive model. The proportion of true positives that are correctly identified as such.
Precision	A metric to measure the performance of a predictive model. The proportion of all predicted positives that are true positives.
F1 score	An overall measure of predictive performance, this is calculated as the harmonic mean of recall and precision.
Deep Learning	A field of machine learning that applies large neural networks that often contain millions of trainable parameters to various predictive tasks.
Hyperparameter	A parameter of a model that cannot be optimized when training the model; it must be selected either at random or through some sort of tuning process prior to training.
Hyperband Algorithm	An algorithm to tune the hyperparameters of a deep learning model. Rather than training a set of random algorithms for a fixed set of iterations, this algorithm will preemptively stop training poorly performing algorithms in favor of algorithms with more promising performance. This saves on computation time and searches a large number of hyperparameter configurations.
Convolutional neural network	A type of neural network originally developed for machine vision tasks, such as image recognition, but has in recent years been shown to work effectively for genomics tasks.
LSTM	Long-Short Term Memory, a type of neural network that analyzes sequential data, often used for tasks such as speech recognition or language translation.
Activation function	A function that is applied to a deep learning layer in order to introduce non-linearity and/or to confine the output to a certain distribution. The sigmoid activation function is often used in binary classification because it limits the final output of the model to a range between zero and one.
Training set	The examples from the dataset used to train the parameters of a machine learning model. High performance on this could indicate overfitting, meaning that the model may not generalize to examples outside of this set.
Validation set	Additional examples that were not used to train the model, but can be used to assess model performance and to tune hyperparameters. Also known as the “dev” set or a “holdout” set.
Test set	A set of examples that were set aside before the hyperparameter tuning and validation process to give an unbiased assessment of the final model’s performance.

Table S1: Glossary of Terms, Related to Figure 1, Figure S1, Figure S2.



		pHMM	DSN1	DSN2	pHMM	DSN1	DSN2	pHMM
		E-value < 1	P(smORF) > 0.5	P(smORF) > 0.5	E-value < 1e-6	P(smORF) > 0.9999	P(smORF) > 0.9999	E-value < 1e-20
Training	Recall	1.000	1.000	1.000	1.000	0.908	0.904	0.901
	Precision	0.909	1.000	0.999	0.995	1.000	1.000	1.000
	F1 Score	0.952	1.000	0.999	0.997	0.949	0.945	0.948
Validation Observed	Recall	1.000	0.990	0.987	0.999	0.824	0.829	0.921
	Precision	0.811	0.988	0.975	0.990	1.000	0.999	1.000
	F1 Score	0.895	0.989	0.981	0.995	0.899	0.900	0.959
Validation Unobserved	Recall	$0.660^{1,4}$	0.647^{4}	0.728^{1}	0.469	0.233	0.360	0.207
	Precision	0.770^{2}	0.858	0.832^{2}	0.982	0.948	0.917	1.000
	F1 Score	$0.710^{3,5}$	0.737^{5}	0.776^{3}	0.634	0.371	0.514	0.342

Table S2: Average recall, precision, and F1 scores for the 64 randomized datasets, Related to Figure 1. Showing all three models with the same significance thresholds described in Figure 1. Superscripts indicates the metrics that were compared and described in the main text using a paired sample t-test. All tests shown here had a $P < 1 \times 10^{-16}$, with the exception of ^{4}, where P = 0.00684.

Table S3: Details of core smORFs identified across 26 bacterial species, Related to Figure 4. Tab “Genome Counts”: Number of genomes analyzed for each species in the core genome analysis. Includes the final 26 species analyzed in the core-genome analysis of this study, and the number of genomes analyzed.

Tab “Genome Identifiers”: A complete list of all genomes analyzed in this study, including the species of origin, the Unique ID (NCBI) of the assembled genome, and an FTP path to the genomic data.

Tab “Core smORFs”: A table summarizing the core smORFs identified in each species' core genome. The column "smORF Cluster" is used internally as an identifier for the smORF cluster. The column "Filtered Set" refers to the filters that were applied to core smORFs in Fig. 4a-c. The column "Median smORF Cluster Length (aa)" refers to the median length of all small proteins found in that species' smORF cluster in amino acids. The column "Proportion of Genomes where smORF Cluster is present" is self-explanatory, a smORF cluster was considered to be "core" if this number exceeded 0.97. The column "Predicted smorfam(s)" refers to the smORF family or families, as identified by the pHMMs, that smORF cluster members are assigned to on at least one occassion. The column "Pfam domain" refers to a specific Pfam domain identified in the smORF cluster. The column "Pfam Domain Type" refers to the labels in Fig. 4A-C. The column "smORF Cluster Example" gives a single sequence example as a representative of the smORF cluster.