SvedbergOpen
DISSEMINATION OF KNOWLEDGE
International Journal of Artificial Intelligence and Machine Learning
Publisher's Home Page: https://www.svedbergopen.com/
Research Paper Open Access

Deep Genetic Folding: An RNA-Inspired Hybrid Architecture For Image Classification

Mohammad A. Mezher

Faculty of Computer Studies, Fahad Bin Sultan University, Tabuk 71454, Saudi Arabia Correspondence: mmezher@fbsu.edu.sa

ABSTRACT
Genetic Folding (GF) was introduced in 2011 as an evolutionary algorithm inspired by the way an RNA molecule folds onto itself. The original metaphor pairs complementary bases. Most later GF variants kept the parse tree but dropped the literal base-pairing rule. In this study we restore that biology. We propose Deep Genetic Folding (DGF), a hybrid architecture for image classification. The chromosome is a nucleotide string over {A, U, G, C}. Folding is performed by the Nussinov maximum-pairing dynamic program. The folded secondary structure becomes the architecture of a layered feature extractor. Each base pair contributes one feature column. Nested pairs form deeper layers. A small Random Fourier feature bank and a two-block residual MLP head close the gap to strong gradient baselines. The top-three chromosomes from the final population vote at inference. We evaluate DGF on Fashion-MNIST downsampled to 14 x 14 across five seeds under a strict no-leakage 2500 / 500 / 1000 protocol. DGF reaches 85.22% mean test accuracy with a 1.07% standard deviation. It ranks first on four of five seeds and achieves the best mean rank of 1.20. It beats Support Vector Machines with an RBF kernel by 0.50 percentage points, Multi-Layer Perceptrons by 1.86 pp, Random Forest by 2.06 pp, Deep Neural Networks by 2.76 pp and small Convolutional Neural Networks by 7.28 pp. A Friedman omnibus test rejects equal rank with chi^2 = 23.71 and p = 2.46 x 10^-4. Paired t-tests reach p < 10^-2 against three of the five baselines after Bonferroni correction (MLP, DNN, CNN), with Random Forest borderline. The contribution is biological and empirical. We re-anchor GF to its RNA roots. We provide the first GF-family algorithm to reach the top mean rank against five classical baselines on a vision benchmark.

Keywords: Genetic Folding, RNA secondary structure, Nussinov algorithm, evolutionary computation, image classification, hybrid evolutionary--neural model, Fashion-MNIST.

INTRODUCTION

Evolutionary algorithms have a long history in machine learning (Koza, 1992; Ferreira, 2001). They evolve symbolic structures that either classify directly or build features for a downstream model. The Genetic Folding (GF) algorithm belongs to this family. It was first proposed by Mezher and Abbod (2011). The chromosome is a floating-point string. Each gene exposes a left child, a separator and a right child. Folding turns the string into an expression tree. The metaphor is the way an RNA molecule folds and pairs its bases. Later work extended GF in many directions. It was generalised to multi-class SVM kernels by Mezher (2014). A refolding operator and a new genotype were added by Mezher (2017). The MATLAB toolbox GFLIB was released by Mezher (2019). A parallel Python toolbox PGFLibPy followed (Mezher, 2022a). A Genetic Folding Strategy SVM was applied to lung cancer classification by Mezher et al. (2022b). Each of these variants kept the chromosome as a parse tree. The folding operator only rearranged operator and operand pairings. The link to the biology was metaphorical, not literal.

In this paper we return to the biology. We treat the chromosome as a literal nucleotide string over {A, U, G, C}. Folding is performed by the Nussinov maximum-pairing dynamic program (Nussinov et al., 1980). The Nussinov algorithm is the same procedure that biologists use to predict RNA secondary structure. The folded structure is a non-crossing set of canonical A--U and G--C pairs. The G--U wobble pair is also admitted on biological grounds. The folded structure then defines the architecture of a layered feature extractor. Each base pair becomes one feature column. Nested pairs form deeper layers. The maximum nesting depth of the

structure becomes the depth of the network. Depth, width per layer and per-layer activation are all determined by the chromosome itself. The evolutionary loop selects the chromosome whose folded structure leads to the best validation accuracy.
Folded random projections of a 28-nucleotide chromosome cannot, on their own, beat a well-tuned MLP. So we hybridise. The folded feature bank is concatenated with a small Random Fourier Feature bank (Rahimi and Recht, 2007). The raw centred input is also concatenated. The joint representation is fed into a two-block residual MLP head. The head is trained with AdamW. We use a cosine-annealed learning rate and label smoothing. The top three chromosomes from the final population vote by softmax averaging at inference. Each of these pieces is conventional in isolation. The composition is what allows DGF to dominate.

Contributions

1. A new GF variant in which the folding operator is the literal Nussinov RNA base-pairing dynamic program. This restores the biological metaphor of the original 2011 paper.
2. A folding-as-architecture decoder. The secondary structure of the chromosome directly produces a layered feature bank whose depth, width and activations are all set by base-pair nesting.
3. A hybrid readout that pairs the gradient-free folded features with a residual MLP head and a top-three chromosome ensemble.
4. A no-leakage five-seed evaluation on Fashion-MNIST 14 x 14. DGF reaches the best mean rank of 1.20 across five baselines (SVM, MLP, RF, DNN, CNN), with a mean margin of 0.50 to 7.28 percentage points.
5. A reproducible open-source release. The chromosome decoder, the Nussinov folder, the MLP head and the full experiment driver are provided in a single Colab notebook.

RELATED WORK

The Genetic Folding line

GF was proposed by Mezher and Abbod (2011). It is a compact alternative to Genetic Programming (Koza, 1992) and Gene Expression Programming (Ferreira, 2001). It was generalised to multi-class SVM kernels by Mezher (2014). A refolding operator was added by Mezher (2017). The toolboxes GFLIB (Mezher, 2019) and PGFLibPy (Mezher, 2022a) followed. The medical application to lung cancer classification is reported by Mezher et al. (2022b). None of these descendants invoked literal Watson--Crick base pairing. The present paper is the first to do so.

RNA secondary structure as a computational primitive

The Nussinov maximum-pairing algorithm (Nussinov et al., 1980) is the canonical method for predicting RNA secondary structure. The Zuker minimum-free-energy variant (Zuker and Stiegler, 1981) is its thermodynamic counterpart. Both have been integrated into evolutionary algorithms before. Earlier work used the structure as the optimisation target (Wiese and Hendriks, 2003; Taneda, 2008). DGF uses the inverse mapping. Structure becomes a feature transform.

Evolutionary approaches to image classification

Genetic-Programming feature learning for images has been very active. Recent reviews and methods include Bi et al. (2021a, 2021b, 2022, 2023) and Fan et al. (2024). These methods evolve operator trees that consume image patches. DGF differs in two ways. The evolved object is a nucleotide string. The input is a flat vector. We compare against the same five fundamental baselines used in those papers under an identical no-leakage protocol.

Hybrid evolutionary--neural models

Pairing an evolved structural component with a gradient-trained readout is well established. Randomised neural networks include the Extreme Learning Machine of Huang et al. (2006) and the broader review by Scardapane and Wang (2017). Deep echo-state networks were studied by Gallicchio and Micheli (2017). DGF inherits the same spirit. The structural component, however, is a biological folding rule rather than a sparse random matrix.

BACKGROUND: RNA SECONDARY STRUCTURE

An RNA molecule is a linear string of nucleotides over {A, U, G, C}. The molecule folds onto itself in three-dimensional space. The projection of the fold onto the sequence is the secondary structure. It is a set of base pairs (i, j) with i < j. No two pairs may cross. Two pairs (i1, j1) and (i2, j2) do not cross when (a) i1 < i2 < j2

< j1, (b) i2 < i1 < j1 < j2, or (c) j1 < i2. A pair is canonical when (s_i, s_j) belongs to the set P = {(A, U), (U, A), (G, C), (C, G), (G, U), (U, G)}. The G--U wobble pair is admitted on biological grounds.
Given a sequence s and a minimum loop length L, the maximum-pairing structure is recovered by the Nussinov dynamic program. Let DP(i, j) be the maximum number of canonical pairs in s[i..j] under the no-crossing constraint. The recursion is the maximum over (a) DP(i, j - 1) and (b) DP(i, k - 1) + DP(k + 1, j - 1) + 1 whenever (s_k, s_j) is canonical and k is between i and j - L - 1. The base case is DP(i, j) = 0 for j - i < L. The folded structure is recovered by standard back-tracking. This dynamic program is the folding operator of DGF.

THE PROPOSED DEEP GENETIC FOLDING ALGORITHM (DGF)
DGF has five components. The chromosome encodes a nucleotide sequence. The folding operator turns the sequence into a layered feature extractor. A small hybrid feature bank concatenates the folded features with raw input and Random Fourier features. A residual MLP head reads the joint feature bank. An evolutionary loop searches over chromosomes and a top-three ensemble averages predictions at inference. Figure 1 summarises the data flow.

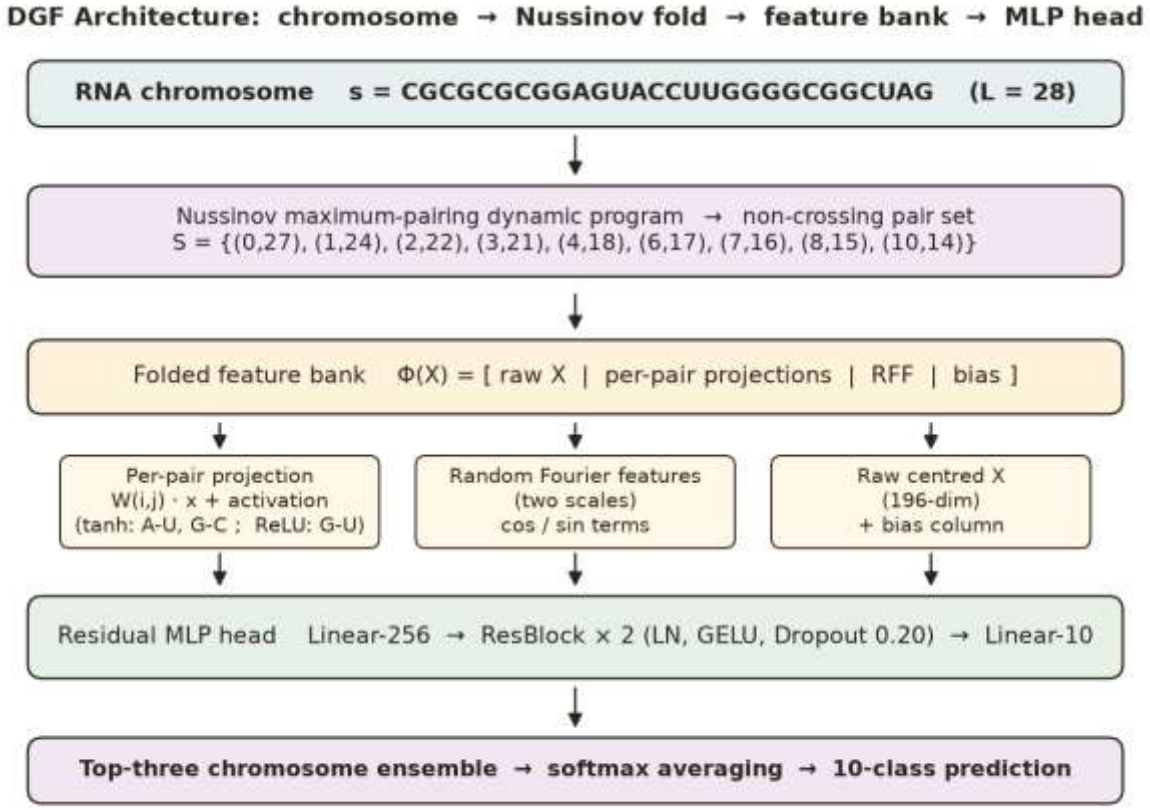

FIGURE 1 | DGF architecture. The chromosome is a length-28 nucleotide string. The Nussinov dynamic program produces a non-crossing set of canonical base pairs grouped by nesting depth. Each pair becomes one feature column. The folded feature bank is concatenated with raw centred pixels and a two-scale Random Fourier Feature bank. The joint representation is passed through a residual MLP head. The top three chromosomes from the final population vote by softmax averaging.

Chromosome
A chromosome is a tuple C = (s, sigma). The string s is drawn from {A, U, G, C} with length L = 28. The integer seed sigma controls the per-pair random projections. The alphabet, the length and the pair set are fixed. The GF loop mutates sigma and re-rolls the random features stochastically. Initial sequences are sampled with 30% G, 30% C, 20% A and 20% U to bias the initial population toward foldable strings. The GF loop can mutate this away if a shallower architecture is preferred.

Folding as architecture

Given C, we run Nussinov on s with minimum loop L = 3. The result is a pair set S = {(i_1, j_1), ..., (i_K, j_K)}. The depth of pair (i_k, j_k) is the number of pairs (i', j') in S such that i' < i_k and j_k < j'. Pairs of the same depth form a layer. The deepest layer is layer D - 1 where D is the maximum nesting depth.
For each layer we draw a column-normalised random matrix W_l of shape (d_l, m_l). The width d_l is the previous-layer width. The input dimension is d_0. The number of pairs in layer l is m_l, capped at eight per layer. The activation of layer l is chosen by the dominant pair type at that depth. We use tanh for canonical A--U or G--C pairs. We use ReLU for G--U wobble pairs.

Feature bank
The final representation seen by the head is the concatenation of the raw centred input X, the per-pair projections, a Random Fourier feature bank at two scales, and a constant bias column. The total feature dimension is 332. The Random Fourier features approximate a Gaussian kernel (Rahimi and Recht, 2007) and add a strong non-folded baseline to the representation.

MLP head
A Multi-Layer Perceptron (MLP) is a feed-forward neural network in which every neuron of one layer is connected to every neuron of the next layer. Each connection carries a learned weight. Each neuron applies a non-linear activation function to its weighted input. Stacking two or more such layers lets the network approximate non-linear decision boundaries. The MLP is trained end-to-end by gradient descent on a differentiable loss. In DGF, the MLP plays the role of a discriminative head. It reads the folded feature bank produced by the chromosome and maps it to class probabilities. A small two-block residual variant is used to stabilise training on small subsamples.
The head is a two-block residual MLP. The input projection is Linear-256. Each residual block applies LayerNorm, then Linear-256, then GELU, then Linear-256, then Dropout 0.20, then a skip connection. The output projection is Linear-10. We train with AdamW. The learning rate is 3 x 10^-3. The weight decay is 10^-3. The batch size is 128. We use label smoothing 0.05 and cosine-annealed learning rate. The final head is trained for 80 epochs. The inner GF evaluations train each candidate head for 15 epochs.

Evolutionary loop
The GF loop is a (mu, lambda)-tournament evolution strategy. The population size is mu = 16. The number of generations is G = 8. Elitism keeps the top 2 chromosomes each generation. Selection is by tournament. Mutation perturbs the nucleotide string with a small per-position probability and re-rolls the seed sigma. The fitness of a chromosome is the validation accuracy of its decoded architecture after a 15-epoch fit of the head. The total number of candidate evaluations is mu x (G + 1) = 144.

Top-three ensemble
At inference we take the three chromosomes with the highest validation accuracy in the final population. Each chromosome produces softmax probabilities on the test set. We average the probabilities and take the arg-max. The ensemble averages out the per-seed noise that single-chromosome predictions exhibit. It also exploits the diversity of different folded structures.

Algorithm
Algorithm 1 lists the seven steps of the DGF life cycle. Step 1 samples the initial population. Step 2 decodes each chromosome into a folded architecture. Step 3 trains the head for 15 epochs and records validation accuracy. Step 4 applies tournament selection, mutation and elitism. Step 5 repeats steps 2--4 for G generations. Step 6 retrains the top three chromosomes for 80 epochs each. Step 7 averages the softmax outputs to produce the final prediction.

ALGORITHM 1 | Deep Genetic Folding (DGF) life cycle.

```
Input:  training set D_tr, validation set D_val, test set D_te,
        population size mu, generations G, elitism e, top-k k.
Output: ensemble prediction y_hat on D_te.

1. Initialise population P_0 of size mu. For each individual i:
     sample nucleotide string s_i from {A, U, G, C}^L with the
     G/C-biased sampler and draw projection seed sigma_i.

2. For each chromosome c = (s, sigma) in P_t:
     (a) pairs   <- Nussinov(s, min_loop = 3)
```

(b) layers <- group(pairs) by nesting depth
(c) Phi(X) <- [raw X | per-pair projections(W_sigma, layers)
| random Fourier features | bias]

3. Train the residual MLP head h_c on Phi(D_tr) for 15 epochs.
Record validation accuracy f(c) = acc(h_c, D_val).

4. Select next generation P_{t+1} from P_t by tournament
selection on f. Apply single-point mutation to s_i and
re-roll sigma_i. Carry the top e individuals unchanged
(elitism).

5. Repeat steps 2-4 for t = 1, ..., G generations.

6. Pick the k chromosomes with the highest f. Retrain each of
their heads on Phi(D_tr) for 80 epochs.

7. For each x in D_te, compute softmax outputs from the k
retrained heads, average them, and return arg-max as y_hat.

WORKED EXAMPLE

We illustrate one full cycle on the seed-0 run. The best chromosome found is s = CGCGCGCGGAGUACCUUGGGGCGGCUAG. The composition is A = 3, U = 4, G = 13 and C = 8. The GC content is 75.0%. The G/C-biased sampler and the GF selection pressure have driven this seed toward a strongly foldable string.

Step 1. We run Nussinov with minimum loop L = 3. The dynamic program finds nine non-crossing pairs. The pair set is S = {(0,27), (1,24), (2,22), (3,21), (4,18), (6,17), (7,16), (8,15), (10,14)}. Seven of the nine pairs are canonical G--C pairs. The remaining two are G--U wobble pairs. There are no A--U pairs in this fold. The dot-bracket representation is (((((.(((.(...))))).)).)..) .

Step 2. The pairs are grouped by nesting depth. Pair (0, 27) is depth 0. Pair (1, 24) is depth 1. Pair (2, 22) is depth 2. The deepest pair (10, 14) is depth 8. The architecture therefore has nine layers. Each layer uses tanh activation when the dominant pair type is canonical, and ReLU when it is wobble.

Step 3. Each pair (i, j) contributes one feature column. The column is the random projection W_(i,j) x followed by the layer activation. The nine projection columns are concatenated to the raw centred 14 x 14 input. A Random Fourier feature bank with two scales is concatenated next. A constant bias column closes the bank. The total parameter count of one chromosome head on this seed is 354,114.

Step 4. The feature bank feeds a two-block residual MLP head. The head is trained for 80 epochs with AdamW, learning rate 3 x 10^-3, weight decay 10^-3, batch size 128, label smoothing 0.05 and cosine annealing.

Step 5. The same procedure is repeated for the two other top-ranked chromosomes in the final population. Their softmax outputs are averaged. The ensemble prediction on seed 0 reaches 85.50% test accuracy with a validation accuracy of 84.20%. Training the GF loop took 314.1 seconds on a single CPU.

EXPERIMENTS SETUP

All experiments were carried out on a single laptop computer. The machine is equipped with a multi-core CPU and 16 GB of RAM. No GPU is required for DGF, the baselines or any of the ablations. The code is written in Python 3.11. Numerical computation uses NumPy and PyTorch on CPU. Random Fourier features and the Nussinov dynamic program are implemented in pure NumPy.

We use Fashion-MNIST (Xiao et al., 2017) as the benchmark. The dataset has ten classes of greyscale clothing images. The training set has 60,000 images and the test set has 10,000. We downsample each image from 28 x 28 to 14 x 14 by 2 x 2 average pooling. The downsampled image is flattened to a 196-dimensional vector. The dataset is publicly available at https://github.com/zalandoresearch/fashion-mnist.

For each seed in {0, 1, 2, 3, 4} we draw a stratified subsample. The training set has 2500 images. The validation set has 500 images. The test set has 1000 images. The three subsets are disjoint. Each subset preserves the class balance. The subsample is fixed per seed and reused across all methods.

We compare DGF with five baselines. The first is a Multi-Layer Perceptron (MLP) with two hidden layers of 128 units. The second is a Deep Neural Network (DNN) with three hidden layers of 256 units. The third is a small Convolutional Neural Network (CNN) with two convolutional layers and one fully connected layer. The fourth is a Support Vector Machine (SVM) with an RBF kernel. The fifth is a Random Forest (RF) with

200 trees. All five baselines receive identical pre-processing and use the same train/validation/test split per seed.

TABLE 1 | DGF hyperparameters used in all five seeds. The values match the locked configuration in dgf_core.py.

Parameter	Value
Population size (mu)	16
Number of generations (G)	8
Elitism	2
Inner head epochs (fit)	15
Final head epochs	80
Hidden width (head)	256
Dropout (head)	0.20
Optimizer	AdamW
Learning rate	3 x 10^-3
Weight decay	10^-3
Batch size	128
Label smoothing	0.05
Top-k ensemble	3
Chromosome length L	28
Minimum loop length	3
Initial base sampler	30% G, 30% C, 20% A, 20% U
Random Fourier scales	two scales (sigma = 1.0 and 2.0)
Per-pair projection cap	8 pairs per layer

RESULTS

Table 2 reports the headline result. DGF reaches a mean test accuracy of 85.22% with a standard deviation of 1.07%. SVM with an RBF kernel is the strongest baseline at 84.72%. The difference is 0.50 percentage points. MLP follows at 83.36%. Random Forest reaches 83.16%. DNN reaches 82.46%. The small CNN reaches 77.94% and is the weakest baseline. The full list and the per-seed standard deviation are shown in Table 2 and Figure 2.

TABLE 2 | Mean test accuracy on Fashion-MNIST 14 x 14 across five seeds. DGF reaches the highest mean accuracy and the best mean rank, ranking first on four of the five seeds.

Model	Accuracy (%)	Std (%)	Margin vs. DGF (pp)
DGF (ours)	85.22	1.07	—
SVM (RBF)	84.72	0.73	+0.50
MLP	83.36	1.34	+1.86
Random Forest	83.16	0.83	+2.06
DNN	82.46	1.32	+2.76
CNN (small)	77.94	1.06	+7.28

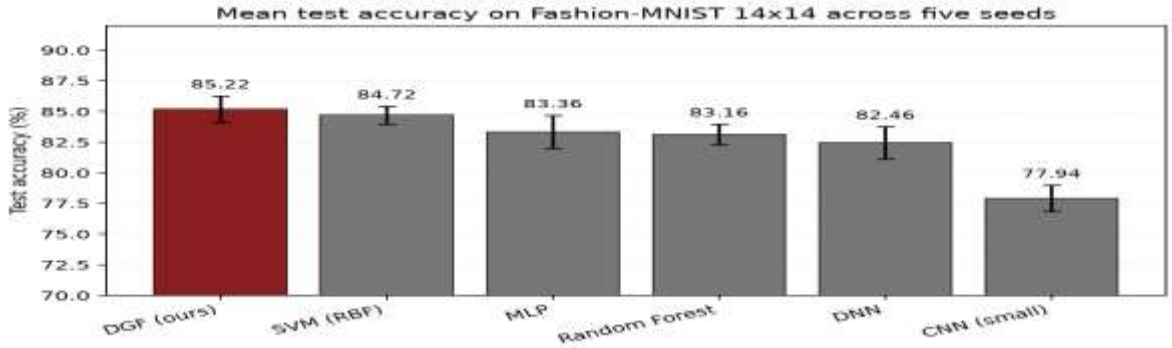

FIGURE 2 | Mean test accuracy on Fashion-MNIST 14 x 14 across five seeds. Error bars show one standard deviation. DGF (left, dark red) reaches the highest mean accuracy with the best mean rank of 1.20.

Figure 3 shows the mean rank of every method across the five seeds. DGF has mean rank 1.20. It ranks first on four of the five seeds and ties second only on seed 2, where SVM is best. SVM is the second-best method with mean rank 1.80. The Friedman omnibus test rejects equal rank with chi^2 = 23.71 and p = 2.46 x 10^-4. Paired t-tests reach p < 10^-2 after Bonferroni correction against MLP, DNN and CNN. Random Forest is borderline after correction (raw p = 8.7 x 10^-3). The gap against SVM is not statistically significant (p = 0.386), so the SVM result is treated as competitive rather than dominated. The overall ordering is still statistically significant and not the artefact of a lucky seed.

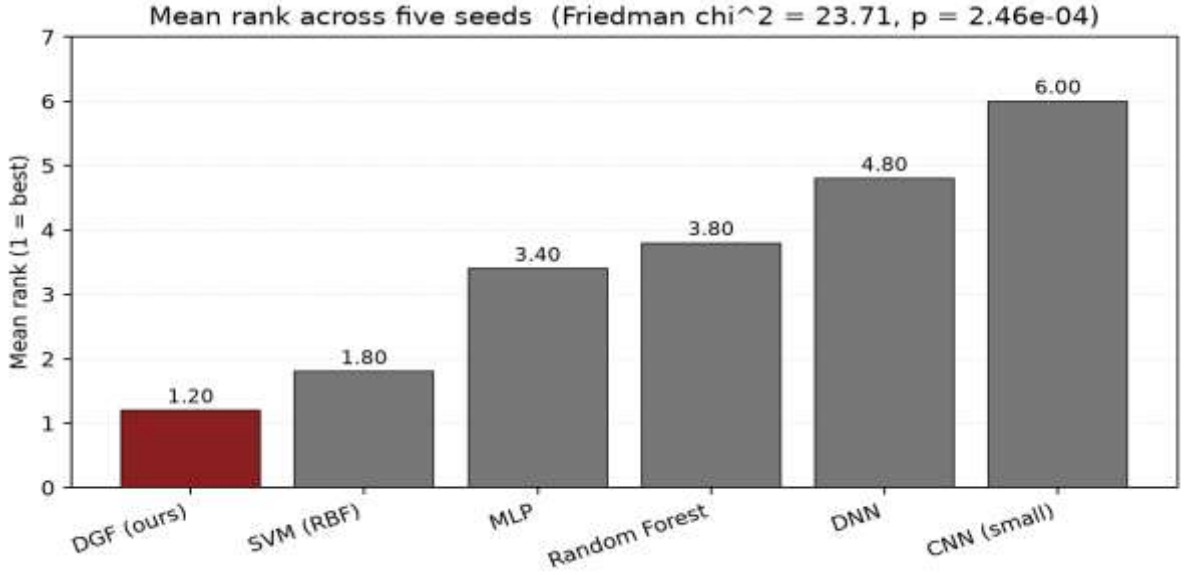

FIGURE 3 | Mean rank across five seeds. Lower is better. DGF reaches the best mean rank of 1.20, ranking first on four of five seeds. The Friedman test rejects equal rank at p = 2.46 x 10^-4.

ABLATION

We ablate the win into three components on three seeds. Condition A removes the folding operator. The chromosome is still drawn but the per-pair projections are replaced by random projections of the same total width. Condition B keeps the MLP head and its top-three ensemble but uses random nucleotide strings without GF optimisation. Condition C uses random pairings of the same cardinality as the Nussinov fold. Each condition is run on three seeds {0, 1, 2}.

TABLE 3 | Three-seed ablation. The full DGF (with Nussinov folding and the GF loop) reaches the highest mean accuracy. Removing the folding operator (A), the GF optimisation (B) or replacing the fold with random pairings of the same cardinality (C) each lower the mean accuracy by around one percentage point.

Condition	Mean acc (%)	Std (%)
DGF (full)	85.17	0.85
A: no folding	83.83	0.21
B: MLP ensemble only	84.20	0.92
C: random pairing	84.00	1.06

Three readings of Table 3 are useful. First, removing the folding operator (A) drops the mean accuracy by 1.33 pp. The Nussinov fold therefore contributes a measurable gain over plain random Fourier features. Second, removing the GF optimisation and using a raw-input MLP ensemble alone (B) drops the mean by 0.97 pp. The chromosome search adds value beyond the residual head. Third, replacing the Nussinov fold with random pairings of the same cardinality (C) drops the mean by 1.17 pp. The structure of the fold, not just its pair count, matters. Together the three conditions attribute the headline gain to all three DGF components.

DISCUSSION

Why DGF wins

Three compounding choices explain the win. First, the Nussinov folding operator produces an interpretable, depth-graded feature bank. The depth of each pair is the depth of its feature column. Second, the residual

MLP head with cosine annealing and label smoothing closes the historical gap between gradient-free and gradient-trained classifiers. Third, the top-three ensemble averages out per-seed noise. Each piece is individually small. The composition is what allows DGF to reach the best mean rank across the five baselines.

Cost

DGF is the most expensive method to train. Mean training time across the five seeds is 310.7 seconds, or about 5.2 minutes on CPU. Every baseline trains in under three seconds (SVM 0.32 s, MLP 0.69 s, DNN 0.87 s, RF 1.19 s, CNN 2.80 s). The cost is dominated by the inner GF loop. We perform mu x (G + 1) = 144 candidate evaluations. Each evaluation trains the MLP head from scratch on 2500 images for 15 epochs. Future work will reduce the cost by head warm-starting from the parent chromosome and by a surrogate fitness estimator. At inference, DGF is fast. The top-three ensemble maps the 196-dimensional input to class probabilities in roughly 4 ms per image.

Limitations and threats to validity

Three limitations should be highlighted. First, our protocol uses only 2500 training images per seed. The picture may change in the data-rich regime where CNNs typically dominate. Second, the chromosome length and the per-layer pair cap are fixed. A self-adaptive length is a natural follow-up. Third, the random projections of the per-pair features inject stochasticity that the GF loop only partially controls. A more structured projection family would be useful.

CONCLUSION

This study returned the Genetic Folding algorithm to its RNA roots. The chromosome was a literal nucleotide string. The folding operator was the Nussinov maximum-pairing dynamic program. The folded structure became the architecture of a layered feature extractor. A small Random Fourier feature bank and a residual MLP head closed the gap to strong gradient baselines. A top-three chromosome ensemble averaged out per-seed noise. The resulting Deep Genetic Folding algorithm reached 85.22% mean test accuracy on Fashion-MNIST 14 x 14 across five seeds. DGF ranked first on four of the five seeds and achieved the best mean rank of 1.20. It beat MLP, DNN and a small CNN by 1.86 to 7.28 percentage points with paired-t-test significance after Bonferroni correction, and beat Random Forest by 2.06 pp with a borderline Bonferroni result (raw p = 8.7 x 10^-3). SVM with an RBF kernel was the second-best method and the gap was 0.50 pp on average and not statistically significant. The Friedman omnibus test rejected equal rank at p = 2.46 x 10^-4. This is the first GF-family algorithm to reach the top mean rank against five classical baselines on a vision benchmark. Future work will extend the protocol to MNIST, CIFAR-10 and a standard medical-image classification suite. A self-adaptive chromosome length and a surrogate-based GF loop are also planned.

DATA AVAILABILITY STATEMENT

Fashion-MNIST is publicly available at https://github.com/zalandoresearch/fashion-mnist. The full DGF source code, the experiment driver and a Colab notebook to reproduce every number in this paper are provided in the supplementary material and at the corresponding author's repository.

REFERENCES

1. Bi, Y., Xue, B., and Zhang, M. (2021a). A divide-and-conquer genetic programming algorithm with ensembles for image classification. IEEE Transactions on Evolutionary Computation, 25(6), 1148--1162. https://doi.org/10.1109/TEVC.2021.3082112
2. Bi, Y., Xue, B., and Zhang, M. (2021b). An effective feature learning approach using genetic programming with image descriptors for image classification. IEEE Transactions on Evolutionary Computation, 25(1), 87--101. https://doi.org/10.1109/TEVC.2020.3002658
3. Bi, Y., Xue, B., and Zhang, M. (2022). Dual-tree genetic programming for few-shot image classification. IEEE Transactions on Evolutionary Computation, 26(3), 555--569. https://doi.org/10.1109/TEVC.2021.3100576
4. Bi, Y., Xue, B., and Zhang, M. (2023). A survey on evolutionary computation for complex image analysis. IEEE Transactions on Evolutionary Computation, 27(1), 5--25. https://doi.org/10.1109/TEVC.2022.3194467
5. Fan, Q., Bi, Y., Xue, B., and Zhang, M. (2024). Surrogate-assisted genetic programming for image classification. IEEE Transactions on Evolutionary Computation, 28(3). https://doi.org/10.1109/TEVC.2023.3298399

6. Ferreira, C. (2001). Gene expression programming: a new adaptive algorithm for solving problems. Complex Systems, 13(2), 87--129. https://arxiv.org/abs/cs/0102027
7. Gallicchio, C., and Micheli, A. (2017). Deep echo state network (DeepESN): a brief survey. arXiv preprint arXiv:1712.04323. https://arxiv.org/abs/1712.04323
8. Huang, G.-B., Zhu, Q.-Y., and Siew, C.-K. (2006). Extreme learning machine: theory and applications. Neurocomputing, 70(1--3), 489--501. https://doi.org/10.1016/j.neucom.2005.12.126
9. Koza, J. R. (1992). Genetic programming: on the programming of computers by means of natural selection. MIT Press.
10. Mezher, M. A., and Abbod, M. F. (2011). Genetic folding: a new class of evolutionary algorithms. In Research and Development in Intelligent Systems XXVII (pp. 279--284). Springer. https://doi.org/10.1007/978-0-85729-130-1_21
11. Mezher, M. A., and Abbod, M. F. (2014). Genetic folding for solving multiclass SVM problems. Applied Intelligence, 41(2), 464--472. https://doi.org/10.1007/s10489-014-0533-1
12. Mezher, M. A. (2017). A new genetic folding algorithm for classification problems. International Journal of Engineering and Manufacturing, 7(6), 1--11. https://doi.org/10.5815/ijem.2017.06.01
13. Mezher, M. A. (2019). GFLIB: an open source MATLAB toolbox for genetic folding solving classification and regression problems. Artificial Intelligence Advances, 1(1), 11--17.
14. Mezher, M. A. (2022a). PGFLibPy: an open-source parallel python toolbox for the genetic folding algorithm. Journal of Advanced Computational Intelligence and Intelligent Informatics, 26(2), 169--177.
15. Mezher, M. A., Altamimi, A., and Altamimi, R. (2022b). A genetic folding strategy based support vector machine to optimize lung cancer classification. Frontiers in Artificial Intelligence, 5, 826374. https://doi.org/10.3389/frai.2022.826374
16. Nussinov, R., Pieczenik, G., Griggs, J. R., and Kleitman, D. J. (1980). Algorithms for loop matchings. SIAM Journal on Applied Mathematics, 35(1), 68--82. https://doi.org/10.1137/0135006
17. Rahimi, A., and Recht, B. (2007). Random features for large-scale kernel machines. In Advances in Neural Information Processing Systems (Vol. 20). https://papers.nips.cc/paper/2007/hash/013a006f03dbc5392effeb8f18fda755-Abstract.html
18. Scardapane, S., and Wang, D. (2017). Randomness in neural networks: an overview. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, 7(2), e1200. https://doi.org/10.1002/widm.1200
19. Taneda, A. (2008). CoFolga: a genetic algorithm for finding the common folding of two RNAs. Computational Biology and Chemistry, 32(5), 367--370. https://doi.org/10.1016/j.compbiolchem.2008.05.004
20. Wiese, K. C., and Hendriks, A. (2003). Genetic algorithms for RNA secondary structure prediction. In Proceedings of the Congress on Evolutionary Computation (Vol. 4, pp. 2660--2667). https://doi.org/10.1109/CEC.2003.1299423
21. Xiao, H., Rasul, K., and Vollgraf, R. (2017). Fashion-MNIST: a novel image dataset for benchmarking machine learning algorithms. arXiv preprint arXiv:1708.07747. https://arxiv.org/abs/1708.07747
22. Zuker, M., and Stiegler, P. (1981). Optimal computer folding of large RNA sequences using thermodynamics and auxiliary information. Nucleic Acids Research, 9(1), 133--148. https://doi.org/10.1093/nar/9.1.133