λ-DNNs and their Implementation in Aerodynamic and Conjugate Heat Transfer Optimization

Marina Kontou, Dimitrios Kapsoulis, Ioannis Baklagis, and Kyriakos Giannakoglou

Parallel CFD & Optimization Unit, School of Mechanical Engineering, National Technical University of Athens, Athens, Greece
kgianna@mail.ntua.gr

Abstract. A fully-connected Deep Neural Network (DNN) architecture, to be referred to as λ-DNN, used to predict 2D/3D scalar fields is presented. In aerodynamics, the λ-DNN is firstly trained on fields computed using a Computational Fluid Dynamics (CFD) software. Then, it can be incorporated into engineering processes in various ways. One possibility is to use them in optimization problems solved by stochastic population-based methods, in which the λ-DNN may act as the surrogate evaluation model, replacing calls to the CFD tool. Another possibility is in multi-disciplinary problems, to replicate the numerical solver for any of the disciplines. This small list of possible usages is not exhaustive and, of course, different usages can be combined. The input to each DNN contains information to identify the geometrical shape and case-related data, nodal coordinates and, in multi-disciplinary problems, interfacing data connecting solvers for different disciplines on adjacent domains. In this paper, the λ-DNN is firstly used in the aerodynamic shape optimization of a wing using evolutionary algorithms, in which it replicates the CFD solver. Then, it is used in a conjugate heat transfer problem dealing with a solid domain in contact with a flow within a duct. In this problem, the λ-DNN acts as a surrogate to the solver of the heat conduction equation on the solid domain, by interfacing with a CFD solver of the fluid domain.

Keywords: Deep Neural Networks · Flow Prediction · Computational Fluid Dynamics · Conjugate Heat Transfer · Evolutionary Optimization of Aerodynamic shapes · Multi-disciplinary Optimization.

1 Introduction

In shape optimization, in one or more disciplines, a number of different designs must be evaluated in order to reach optimal solutions. This number increases when a stochastic optimization method is used so as to avoid being trapped into a local minimum. In multi-disciplinary optimization, such as a Conjugate Heat Transfer (CHT) one, the analysis of any configuration requires the iterative

solution of the flow equations over the fluid domain and the heat conduction PDE over the adjacent solid domain; during the iterative solution, the two domain solvers communicate at their interface by exchanging heat flux and temperature distributions. The repetitive calls to the two solvers make the cost of a CHT analysis high enough. As industrial requirements become more demanding, not only the number of evaluations needed to reach the optimal solution but, also, the cost per evaluation must be kept as low as possible. This is the main purpose of this paper and the main reason for developing the proposed λ-DNN.

Deep Neural Networks (DNNs) [6] are known to effectively replicate complex tasks by recognizing their core features. As such, in view of the previous discussion, DNNs should learn how to predict the outcome of simulation codes solving PDEs modeling physical phenomena or only the integral quantities needed to evaluate the quality of a configuration, as it may suffice in most optimization problems. In a CHT or any other multi-disciplinary problem, a good idea is to make them replicate the solver corresponding, for instance, to one of the involved disciplines. This might be good enough to shorten the wall clock time of such a simulation. For instance, in an optimization loop, the expected gain from such an algorithm must be evaluated by considering the cost for performing runs to collect the necessary training patterns and that for training the DNN together with the expected reduction in the cost of the optimization loop involving DNNs. This paper demonstrates the efficiency of the proposed optimization techniques based on the newly developed DNNs.

The proposed DNN, which will be referred to as the λ-DNN, fig. 1, is a Fully-Connected Neural Network (FCNN), compact and with a relatively small number of trainable parameters. This is described in detail in Section 2.1.

It is not the first time DNNs are used for predicting flow fields. Convolutional Neural Networks (CNNs), which consist of convolutional layers with local connections between neurons of successive layers, have been used to predict aerodynamic flow fields in unseen flow conditions and geometries, [4]. U-Networks have been used for the prediction of incompressible laminar flows, [5]. They consist of an encoder which compresses the information with successive convolutional layers and a decoder which decompresses it with deconvolution layers and yields the output. In [9], a hybrid DNN, formed by CNN, convolutional Long Short Term Memory network (ConvLSTM) and decoding-CNN (D-CNN), is used for the prediction of unsteady flows.

In this paper, the λ-DNN architecture is applied in the aerodynamic shape optimization of a wing and that of a fluid and a solid domain in a CHT problem, using evolutionary algorithms occasionally assisted by on-line trained metamodels (Radial Basis Function, RBF) networks. The λ-DNN is capable of predicting physical quantities (flow fields in CFD, temperature distributions in CHT) on any type of computational grids (structured or unstructured), since grid connectivity is not required. The λ-DNN requires a small number of training fields and has a low computational cost (compared to [4, 5]), ensuring acceptable accuracy.

2 Methods and Tools

2.1 The proposed λ-DNN

A DNN consists of many layers of neurons, with weights and biases being the network parameters to be computed during the training phase. During this phase, a cost function expressed as the root-mean-squared (RMS) of the differences between the network outputs and the known/archived responses is minimized. To this end, the back-propagation algorithm, [15], using closed-form expressions for the derivatives of the cost function with respect to the network parameters, is used.

The λ-DNN is exclusively formed by fully-connected layers. For better data handling, its architecture consists of two separated branches (fig. 1), one for each type of input. Each branch first passes through fully-connected layers and then connects to the main branch which further processes signals through its own fully-connected layers and concludes to the network output(s). In the cases presented below, there are two different kinds of inputs, one refers to the mesh (nodal coordinates) and the other to parameterization or physical quantities. The different branches allow separate processing of input data of different nature. This leads to extraction of different features for each branch, before merging them to produce the final output.

The Rectified Linear Unit (ReLU) activation function [8] is used in all but the last layer which uses the sigmoid function, for the one-discipline case, and the Hyberbolic Tangent function (tanh) for the two-discipline one. The pre- and post-processing of the set of training patterns and all DNN phases (construction, training and testing) have been implemented in Python 3.6 [14] linked with TensorFlow [1] and run on both GPUs and CPUs.

The λ-DNN is applied according to two different modes. The first is based on a node-to-node logic, meaning that the network predicts only one output, namely the flow variable at each grid node, while the second processes the whole grid at once and the output of a λ-DNN run is a whole field. The difference between the two modes lies on whether grid connectivity is taken into account; the former is independent of the grid connectivity while the latter is not.

2.2 Computational Tools for Aerodynamic and CHT Analyses

For the numerical solution of CHT problems, [2, 12], coupled and decoupled solution schemes are in use. In the coupled approach, PDEs on different domains are simultaneously solved while, in the decoupled approach, each discipline is solved separately and provides boundary conditions for the others by exchanging information along their interface. In either approach, proper solvers' interfacing guarantees heat flux conservation and equal temperatures over the fluid-solid interface. Herein, the decoupled approach is used. The procedure is computationally expensive as it requires many iterative cycles (CHT cycles), with calls to the CFD solver (for the fluid domain) and the Heat Conduction one (for the solid domain). Each domain is solved separately and the interaction of the

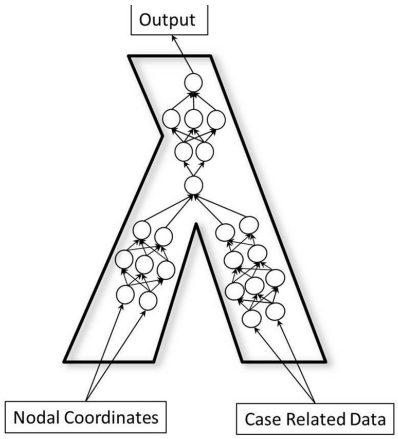

Fig. 1. The proposed λ-DNN architecture. The DNN is named after its shape that looks like the Greek letter λ. For visualization reasons, each circle/layer comprises a number of neurons/layers. The number of inputs/outputs and that of fully-connected layers vary among the cases.

solvers is taking place at their interface, so a DNN trained on the contribution of the one solver to the interface can be very helpful. By doing so, the cost of one solver will be significantly reduced, together with the overall cost of the CHT evaluation. More details about this implementation and the offered gains will be provided in section 4, where the corresponding case is described.

Regarding the analysis of fluid flows, an in-house Reynolds-Averaged Navier-Stokes equations' solver, [3, 10], coupled with a Heat Conduction equations' solver is used. The governing equations are discretized using the finite volume technique. The CFD code solves incompressible and compressible flows; herein, all cases are studied with the compressible variant. Fluid and solid domain solvers fully exploit the CUDA programming environment and run on a GPU cluster. CFD/CHT evaluations and DNN training were performed on NVIDIA K20 or K40 GPUs.

2.3 EA-based Optimization, without or with Metamodels

For the EA-based optimizations presented in sections 3 and 4, the optimization platform EASY, [7], developed by the Parallel CFD & Optimization Unit of the National Technical University of Athens is used. Real encoding of the design variables, with 5% mutation probability and simulated binary crossover with 90% probability are used. To reduce the cost of the EA-based optimization, EASY optionally employs on-line trained metamodels (RBF networks) and this is performed in a rather unique way compared to other similar tools. The role of the metamodel(s) is to replace calls to the problem-specific method (PSM; herein, either a CFD or a CHT solver) by approximating the objective function(s) value(s) at negligible cost, after training them on data collected for individuals already evaluated on the PSM. The distinguishing feature of the Metamodel-Assisted EA (MAEA) implemented in EASY is that the RBF networks training occurs during the evolution (this is what "on-line" stands for). The use of metamodels starts after the first T^{MM} individuals have been evaluated on the PSM and

recorded in the MAEA database (DB). In all subsequent generations, a different RBF network per individual is trained on a few selected neighboring DB entries. Using these RBF networks, all population members are pre-evaluated and only the most promising among them (λ_e; a small user-defined value per generation) are re-evaluated on the PSM, [11] and recorded in the DB.

This is contrasted to the most widely used EAs supported by off-line trained metamodels, which usually rely on a single metamodel, valid over the whole search space. This metamodel is trained with evaluated individuals resulting from a Design of Experiments (DoE) technique, [13]. In such an algorithm, the EA–based search relies exclusively upon the previously (off-line) trained metamodel. The "optimal" solution(s) is/are re-evaluated on the PSM and the process goes on by re-training the metamodel on the updated DB and performing a new EA-based search until a termination criterion be met.

In this paper, in the sake of pluralism, both on- and off-line metamodels are used within EASY. The λ-DNN is used as off-line metamodel, replacing the CFD tool. The λ-DNN is updated if, upon convergence of the EA-based search using the same λ-DNN, the re-evaluation of the "optimal" solution (quotes denote the outcome of an optimization in which the evaluation tool is a surrogate model) on the CFD asks for better accuracy. In the CHT problem, the evaluation s/w combines an accurate tool (CFD solver) for the fluid domain and the λ-DNN for the solid domain. This optimization also uses on-line trained metamodels (a MAEA with RBF networks, implemented as described above).

3 Case I: Aerodynamic Shape Optimization

The first application is concerned with the use of the λ-DNN in the prediction and the aerodynamic shape optimization of a transonic wing. The geometry of [16] is the reference wing. The flow is inviscid with free-stream Mach number and flow angles equal to $M_\infty = 0.8395$, $a_{\infty,pitch} = 3.06^o$ and $a_{\infty,yaw} = 0^o$. The wing shape is encapsulated within a $6 \times 3 \times 3$ volumetric NURBS (trivariate Non-Uniform Rational B-Splines) control grid. A knot vector and a degree per parametric direction are needed to complete the parameterization. The morphing used in this case is graphically presented (as a 2D example, though) in Case II. 12, out of the 54 control points (CPS), are allowed to move in the chordwise and the normal-to-the-planform directions within ±20% of their reference coordinate values, resulting to $12 \times 2 = 24$ design variables in total. In this case, the parameterization is used not only for generating training patterns but, also, as an input to the DNN to identify different wing shapes. An unstructured 3D CFD grid of $\sim 1.33 \times 10^6$ nodes generated around the reference geometry and adapted to all changed shapes is used. A single run of the CFD software takes $\sim 2min$ on a K20 GPU.

Both a λ-DNN and an FCNN are used. The first branch of the λ-DNN consists of four fully-connected layers with $128, 256, 256, 128$ neurons each; the second branch consists of three fully-connected layers with $128, 256, 128$ neurons each. After their merging, another three fully-connected layers with $128, 256, 128$

neurons each lead to the final output. For comparison reasons the number of neurons and layers in the λ-DNN and the FCNN are kept the same. The FCNN consists of seven layers with $256, 512, 512, 256, 256, 512, 256$ neurons, respectively. The 200 wings are used for training the λ-DNN and the FCNN by randomly changing the position of the pre-defined CPs of the morphing box. In this case, a node-to-node procedure is followed.

The networks are trained on the $M \simeq 27000$ surface nodes of each CFD grid, thus the total number of training patterns is 200×27000. The input data to the networks are the 24 coordinates of the free-to-move control points along with the (x, y, z) coordinates of any surface node, i.e. 27 inputs in total, and the output is a single pressure value at this surface node. The first branch of the λ-DNN processes the coordinates of the control points, while the second one the (x, y, z) coordinates.

In this case, the training cost is $\sim 1.2h$ for both networks. Before proceeding with the shape optimization, the prediction quality of the DNNs on two new wings generated by displacing the control points within the aforementioned bounds, and not seen by the DNNs before, is checked. Predictions of pressure (p_{DNN}) using the trained DNNs are compared with CFD evaluations (p_{CFD}) using the Mean Absolute Percentage Error ($MAPE_p$) indicator per wing, given by:

$$MAPE_p = \frac{1}{M} \sum_{i=1}^{M} \left| \frac{p_{i,CFD} - p_{i,DNN}}{p_{i,CFD}} \right| \tag{1}$$

The average $MAPE_p$ computed on the two new wings is equal to 0.81% for the λ-DNN and 3.83% for the FCNN; and this demonstrates the superiority of the proposed network type.

Then, two EA-based shape optimizations, for maximum wing lift (L), are performed. During the first one, the λ-DNN assists the EAs to reduce the total number of calls to the expensive CFD s/w, by acting as an off-line trained surrogate (Run_1). The λ-DNN predicts the pressure field on the surface of each new wing presented to it which, through integration, computes the lift force. A CFD run on the "optimal" solution is performed only at the end of the EA-based cycle. The second one is a MAEA (with on-line trained RBF networks, $T^{MM} = 40$ and $\lambda_e = 3$) optimization relying upon the CFD code instead of the λ-DNN (Run_2). A comparison between Run_1 and Run_2 is made. Both are configured with 10 parents and 20 offspring, i.e. they are (10,20) EAs or MAEAs. This is a well performing set-up of the optimization method in this case which was kept the same in both cases, in the sake of fair comparison.

For either run, a total budget of 250 CFD evaluations ("time-units") is a priori defined. For Run_1, this budget corresponds to: 200 time-units to form the DB, 33 to train the network and only 17 for running the optimization, spent, in particular, for the necessary CFD re-evaluations. Run_1 resulted to a wing geometry having a lift that is by 2.9% higher than the solution of Run_2. We conclude that the usage of the λ-DNN improves the efficiency of the EA-based search even more than the (already very fast) MAEA. In fig. 2, the convergence histories of the two optimizations are illustrated. To quantify the gain with

respect to the original shape, L is always dimensionalized by the reference wing L value.

In fig. 3, the pressure field for the optimal geometry is shown for the solutions of Run_1 and Run_2. For the former, two pressure fields are presented, that evaluated by the CFD and that predicted by the λ-DNN. The pressure fields differ between Run_1 and Run_2, since the two optimizations resulted in different wing shapes. For the optimized wing of Run_1, the pressure prediction of the λ-DNN (fig. 3, right) is so close to the CFD solution (fig. 3, middle), and this demonstrates the reliability of the proposed network.

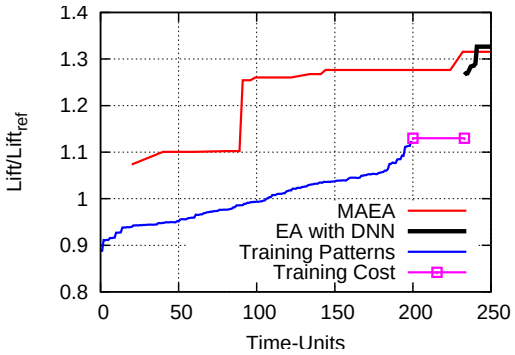

Fig. 2. Case I: Convergence History of the EA relying on the λ-DNN (Run_1) and the MAEA (based on on-line trained RBF networks) (Run_2). The blue line corresponds to the 200 training patterns plotted in ascending order, the pink line represents the training cost, while the black line represents the EA relying on the λ-DNN; all three of them stand for Run_1.

4 Case II: CHT Shape Optimization

Here, the λ-DNN is used in the CHT optimization of a fluid duct adjacent to a solid domain, to replace the heat conduction equation solver on the solid domain. The same CFD code for the fluid domain (this time solving the Reynolds-Averaged Navier-Stokes equations with the Spalart-Allmaras turbulence model, [17]) is used. In the absence of the λ-DNN, the CFD code interacts with a solver for the conduction equation over the solid domain. The same morphing technique (adapted to 2D cases) is used. The high temperature solid domain is cooled by fluid of lower temperature (T) flowing within the S-bend duct. The two domains are solved in a decoupled manner, by interchanging the computed heat flux (from the fluid to the solid) and temperature (in the opposite direction) distributions over the fluid-solid interface (FSI).

For the fluid domain, the flow has inlet total pressure $p_{t,inlet} = 105kPa$, inlet total temperature $T_{t,inlet} = 300K$, outlet static pressure $p_{outlet} = 100kPa$. For the solid domain, a constant temperature of $T = 500K$ is imposed along its bottom

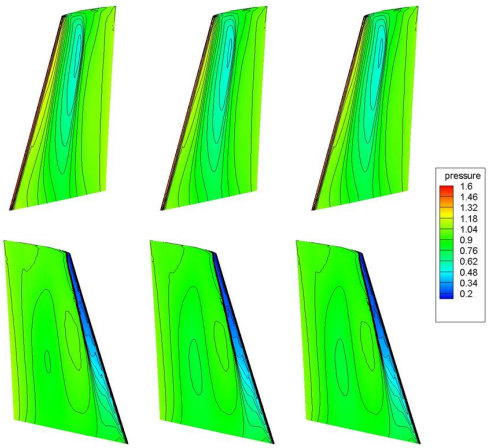

Fig. 3. Case I: Pressure Fields on the surface of the transonic wing. Left: Optimized solution of Run_2. Middle: Optimized solution of Run_1, evaluated by CFD. Right: Optimized solution of Run_1, re-evaluated by the λ-DNN.

boundary, while the rest non-FSI nodes are considered adiabatic. The geometry is parameterized using a 8×6 volumetric NURBS control grid, fig. 4. 24 CPs are allowed to move within $\pm 10\%$ of their reference coordinate values in either direction, resulting to $24 \times 2 = 48$ design variables in the optimization problem. In this 2D case, the parameterization is common for both the fluid and the solid domain but it is not used as input (information) to the DNN. A mono-block structured grid of ~ 330000 nodes is used to discretize both domains, where $M = 240000$ of them correspond to the solid domain. A single run of the CHT solver takes $\sim 20min$ on a K40 GPU.

The database for training the λ-DNN is built by evaluating 180 geometries by randomly moving the CPs. As in the previous case, a λ-DNN and an FCNN are used. The two branches of the λ-DNN consist of one layer each with 512 neurons. The two branches merge to a single layer of 20 neurons that leads to the final output. The network architecture is selected after a trial-and-error procedure. The FCNN consists of two layers with 1024 and 512 neurons each.

The input data are the (x, y) coordinates and the heat fluxes at the 600 FSI nodes, i.e. 1800 inputs in total. At the λ-DNN, the first branch processes the (x, y) coordinates, while the other the heat fluxes. The output is the temperature (T) field on the entire solid domain; grid connectivity and number of nodes remain the same for all the geometries. It is important to notice that, in contrast to the previous case where a node-to-node procedure was followed, here, the nodal T values over the entire solid domain are simultaneously predicted. Two new geometries, not seen by the DNNs before, are used to validate the λ-DNN and the FCNN. In this case the average values $MAPE_T$ 0.3% for the λ-DNN and 0.8% for the FCNN are computed. The λ-DNN has better prediction accuracy and, thus, this is used during the optimization.

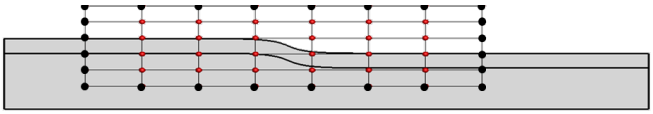

Fig. 4. Case II: Control Points of the volumetric NURBS, morphing box, parameterizing the S-bend duct (upper) and a small part of the solid domain (lower). Black points are kept fixed, whereas red ones are allowed to vary in either direction, morphing the enclosed parts of the fluid and solid domains.

Three MAEA-based optimizations are performed with 10 parents and 20 offspring, supported by on-line trained RBF networks with $T^{MM}=40$ and $\lambda_e=3$.

The first two optimizations aim at minimizing the mass-averaged total pressure losses between the inlet (I) and the outlet (O) of the S-bend duct, i.e. they are dealing with an objective defined only in the fluid domain,

$$F_1 = \frac{\int_{S_I} p_t \rho v_n dS + \int_{S_O} p_t \rho v_n dS}{\int_{S_I} \rho v_n dS} \tag{2}$$

ρ, v_n stand for the density and the normal (directed outwards) to the boundary velocity component.

The first optimization (Run_1) uses the λ-DNN as surrogate to the solid domain solver, while the second one (Run_2) is based exclusively on the solution of the governing PDEs. A total budget of 250 CHT evaluations ("time-units"; one time-unit is the cost per evaluation in Run_2) is decided for both of them. For Run_1, this budget corresponds to 30 time-units to form the database, 24 for training the network and 196 for the optimization, since an evaluation on the (CFD and λ-DNN) tool costs about 0.76 time-units. Then, Run_2 is performed. Fig. 5 presents the convergence histories of the MAEAs and the resulting optimal geometry; a reduction in F_1 ∼8.6%, compared to the initial/reference geometry, is obtained in both Run_1 and Run_2. F_1 is normalized by the value of the same quantity in the reference geometry. The optimal solution of Run_1 is re-evaluated on the exact CHT solver; the λ-DNN error is less than 0.008% verifying the network accuracy.

Then, a two-objective optimization, by adding a second target defined over the solid domain, is performed. The second objective function is the percentage of the area of the solid domain over which T exceeds a threshold value, scaled by the excess temperature, namely

$$F_2 = \frac{1}{\Omega_s} \int_{\Omega_S} (T - T_{thres}) d\Omega \tag{3}$$

F_2 should be minimized. The $T = 500K$ value along the bottom boundary of the solid domain is the highest temperature over the domain. The value of the $T_{thres} = 450K$ is defined in order to minimize the area of the domain which exceeds this threshold value. In whatever follows, F_1 and F_2 are presented in a non-dimensional form; they are both divided by the corresponding values of the reference solution (the one presented in fig. 4).

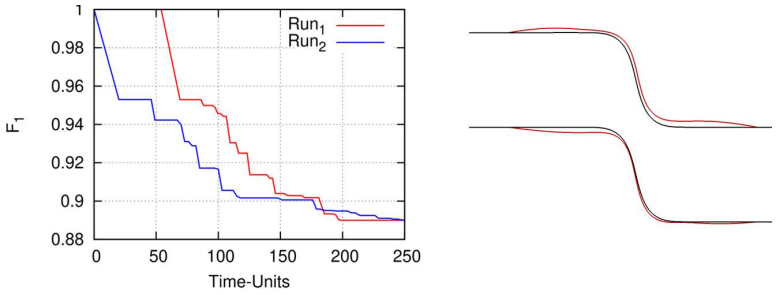

Fig. 5. Case II: Left: Convergence histories of the MAEA-based optimizations for Run_1 (Fluid-(λ-DNN)) and Run_2 (Fluid-Solid). Right: The initial (black) and the optimal (red) geometries (x and y axes not in scale).

The new optimization (Run_3) uses the λ-DNN as the surrogate to the solid domain solver. The total budget is still 250 CHT evaluations distributed as for Run_1. In fig. 6, the front of non-dominated solutions resulted from Run_3 is presented. The members of the front are re-evaluated with the exact CHT solver to verify the network accuracy. The re-evaluated members remain non-dominated and the errors in F_1 and F_2 are $\sim$0.008% and $\sim$0.09%, respectively. The T field from Run_3 and its re-evaluation as well as the prediction error are shown for the member of the front with $F_1 = 0.975$ and $F_2 = 0.997$ in figs. 7 and 8, respectively. As shown in figs. 6 to 8, the accuracy of the λ-DNN is high enough to sufficiently replicate the heat conduction equation solver on the solid domain during an optimization and reduce its overall cost.

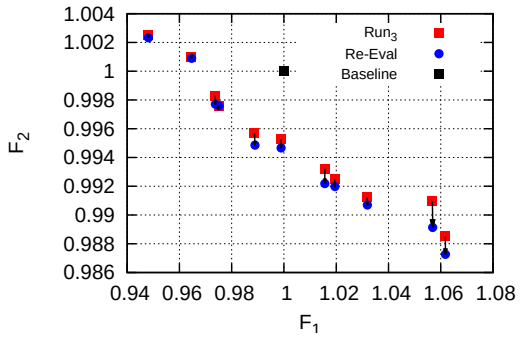

Fig. 6. Case II: Front of non-dominated solutions computed from Run_3 (red squares), plotted along with the baseline geometry (black square). The front members are re-evaluated on the exact CHT solver resulting to a new non-dominated front (blue circles). The computed "optimal" members on the front remain non-dominated after the re-evaluation using the computational mechanics s/w.

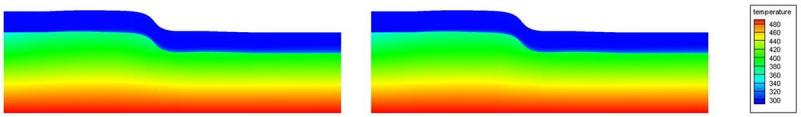

Fig. 7. Case II: Temperature field as resulted from Run_3 (left) and re-evaluated with the exact CHT solver (right) for the member of the front with $F_1 = 0.975$ and $F_2 = 0.997$ (x and y axes not in scale).

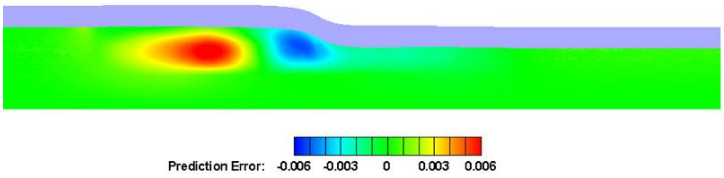

Fig. 8. Case II: Prediction error of the solid domain temperature field for the member of the front with $F_1 = 0.975$ and $F_2 = 0.997$.

5 Conclusions

This paper presented a new DNN architecture, called λ-DNN, which can be used in single- and multi-disciplinary optimization. The λ-DNN is used to predict CFD and CHT results in the form of entire flow fields or temperature distributions, so as to replace expensive runs of CFD/CHT solvers in design/optimization processes. The proposed method has been demonstrated in a 3D one-discipline and a 2D two-discipline case. The DNN yields very good predictions of fields. Inputs are presented to two different branches for better and separate processing of the given input data and, then, these two branches meet at a single one that gives the required output. The λ-DNN manages to reduce the training cost and number of patterns required. The high prediction accuracy of the proposed λ-DNN was demonstrated in a wing flow problem and a solid domain adjacent to a S-bend duct as a surrogate to the heat conduction solver. Regarding the one-discipline case optimization, the trained λ-DNN is used as metamodel during an EA-based optimization and improves the search performance with results comparable with those provided by the MAEA. Regarding the two-discipline case, the λ-DNN is used instead of the Heat Conduction solver during a MAEA optimization. Therefore, the proposed λ-DNN is shown to be capable of replacing the CFD and CHT solvers in expensive (analysis and optimization) processes. Future work includes the use of the λ-DNN in 3D real-world one- and two-discipline applications, such as aeroelastic shape optimization.

6 Acknowledgments

This work has been supported by the Greek Research and Technology Network (GRNET) High Performance Computing Services, through the TurboNN and CGT-DNN projects.

References

1. Abadi, M., Agarwal, A., Barham, P.: TensorFlow: Large-scale machine learning on heterogeneous systems (2015), http://tensorflow.org/, software available from tensorflow.org
2. Andrei, L., Andreini, A., Facchini, B., Winchler, L.: A decoupled cht procedure: application and validation on a gas turbine vane with different cooling configurations. 68th Conference of the Italian Thermal Machines Engineering Association (2013)
3. Asouti, V., Trompoukis, X., Kampolis, I., Giannakoglou, K.: Unsteady CFD computations using vertex-centered finite volumes for unstructured grids on Graphics Processing Units. International Journal for Numerical Methods in Fluids **67**(2), 232–246 (2011)
4. Bhatnagar, S., Afshar, Y., Pan, S., Duraisamy, K., Kaushik, S.: Prediction of aerodynamic flow fields using convolutioanl neural networks **64**, 525–545 (2019)
5. Chen, J., Viquerat, J., Hachem, E.: U-net architectures for fast prediction of incompressible laminar flows (2019)
6. Deng, L., Yu, D.: Deep learning: Methods and applications. Foundations and Trends in Signal Processing **7**(3-4), 1–199 (2014)
7. Giannakoglou, K.: The EASY (Evolutionary Algorithms SYstem) software, http://velos0.ltt.mech.ntua.gr/EASY. (2008)
8. Goodfellow, I., Bengio, Y., Courville, A.: Deep Learning. The MIT Press (2016)
9. Han, R., Wang, Y., Zhang, Y., Chen, G.: A new prediction method of unsteady wake flow by the hybrid deep neural network (2019)
10. Kampolis, I., Trompoukis, X., Asouti, V., Giannakoglou, K.: CFD-based analysis and two-level aerodynamic optimization on graphics processing units. Computer Methods in Applied Mechanics and Engineering **199**(9-12), 712–722 (2010)
11. Karakasis, M., Giotis, A., Giannakoglou, K.: Inexact information aided, low-cost, distributed genetic algorithms for aerodynamic shape optimization. International Journal for Numerical Methods in Fluids **43**(10-11), 1149–1166 (2003)
12. Moretti, R., Errera, M.P., Couaillier, V., Feyel, F.: Stability, convergence and optimization of interface treatments in weak and strong thermal fluid-structure interaction. international Journal of Thermal Sciences **126**, 23–37 (2017)
13. Myers, R., Montgomery, D.: Response Surface Methodology Process and Product Optimization Using Designed Experiments. 2nd edn. (2002)
14. Rossum, G.: Python tutorial. Technical Report CS-R9526 (1995)
15. Rumelhart, D., Hinton, G., Williams, R.: Learning representations by back-propagating errors. Nature **323**(6088), 533–536 (1986)
16. Schmitt, V., Charpin, F.: Pressure distributions on the ONERA M6 wing at transonic mach numbers, experimental data base for computer program assessment. Tech. rep., AGARD 138 (1979)
17. Spalart, P., Allmaras, S.: A one–equation turbulence model for aerodynamic flows. La Recherche Aerospatiale **1**, 5–21 (1994)