Neural Feature Learning for Engineering Problems

Xiangxiang Xu, Lizhong Zheng, Ishank Agrawal

***Abstract*—Using deep neural networks as elements of engineering solutions can potentially enhance the overall performance of the system. However, most existing practices that use DNNs as black boxes make integrating DNN modules in engineering systems hard. In this paper, we address one of such difficulties: in engineering solutions, we often look for parameterized solutions that perform well in a collection of scenarios. In most problems, this means the DNN modules are trained and used in different environments. Instead of using a transfer learning or multi-task learning formulation, which are common in the literature, we argue that such problems are intrinsically about the multi-variate dependence between the data, the label, and the environment parameters. Using an example of symbol detection over wireless fading channels with interference, we demonstrate that such problems can generally be solved as a modal decomposition. We develop new metrics to measure the information contents of features and some basic neural network architectures to perform geometric operations in the space of feature functions. With these building blocks, we discuss the steps to build a receiver that does not require any online training but can adapt to different fading scenarios when given the channel state information (CSI). We use the symbol detection problem as an example to discuss some key issues and steps to include DNN modules in complex engineering systems.**

I. Introduction

Deep neural networks (DNN) and, more generally, learning-based techniques are natural choices to extend our solutions to engineering problems. In complex engineering systems, conventional model-based analytical solutions often find realistic models with non-linear, non-stationary, and non-Gaussian elements intractable. In such cases, DNN modules, which can approximate non-linear functions and have powerful standardized implementations, can be good options for extending the functionality or improving the performance. One such example is the current initiative of AI-native solutions for 5G/6G communication systems [1].

There are, however, some fundamental difficulties in including DNN modules as parts of engineering solutions. Most existing practices use DNNs as black boxes, which cannot offer performance guarantees commonly required in engineering problems, often require extensive computational power and training time, and, in most cases, cannot effectively incorporate the domain knowledge about the specific applications.

In this paper, we address one of the issues in using DNNs in engineering problems: the neural network is often trained in a scenario different from where it is deployed.

We will use one specific symbol detection problem in fading channels as an example to illustrate our key ideas. In this problem, our goal is to determine the transmitted symbol from the received signals, which is a typical classification setup. However, the wireless channel can be affected by several factors: the transmitted signal power, the modulation scheme, the propagation loss, the fading environment, etc. The variation of the environment is characterized by a set of commonly used parameters, including the SNR and the fading coefficients. These parameters are collectively known as channel state information (CSI), which is known to the receiver through control channel coordination or an online pilot estimation procedure. Furthermore, the statistical distributions of these varying parameters are well-known and specified in wireless communication standards. Thus, the symbol detector we need to design has both the received signals and the CSI parameters as the input. Such situations where we need parameterized processing are very common in engineering applications. This paper will develop the general concepts and procedures required to address such problems by focusing on the symbol detection problem as an example.

The challenge of the above problem is that the ideal receiver behaves rather differently as the channel parameters change. A brute-force solution is to train a neural network with both the received signals and the parameters as inputs. This is often unsatisfactory since ensuring the neural network follows the parameter to cover all possible cases is difficult, resulting in oversized DNNs and high training costs. In practice, it is more common that the neural network is first trained offline with data generated from one fading case or mixed samples from a collection of cases. When the system is put online, a smaller set of online samples is used to adapt the neural network to the target scenario. This can be viewed as a transfer learning approach. Since the online samples are often expensive, the adaptability of this method is limited.

In this work, we view the core issue of the receiver design problem as learning a 3-way dependence between the received signal, the transmitted symbol, and the channel parameter unknown at the training time. We establish a framework for explicitly learning features from data samples to capture this statistical dependence. To this end, we introduce a geometry on functional space. We then formulate the feature learning problem as finding a good low-rank approximation of the dependence model, with approximation error measured by the distance in the functional space. We then develop a nested DNN architecture to perform basic geometric operations in feature function space, such as making orthogonal projections. This method allows us to use separate neural network modules to learn informative features of the received signals that are parameter-invariant and parameter-dependent. We then can assemble these learned modules to form the optimal receiver. The key advantage of our method is that it separates the

*This research is supported by ONR Grant N00014-19-1-2621

X. Xu (xuxx@mit.edu), L. Zheng (lizhong@mit.edu), and I. Agrawal (ishank@mit.edu) are with the Dept. EECS at MIT.

dependence on the channel parameters. The parameter value explicitly controls the receiver behavior without requiring any online training. Conceptually, if we understand DNN as learning statistical models with the internal weights as parameters, then our approach forces some of these parameters to be aligned with the channel parameters that humans use to characterize different environments, such that our knowledge of the value of the channel parameters can be used without re-estimated.

In the following, we will first, in Section II, establish the mathematical structure of feature functional space and its relation to neural network approximations. In particular, in II-E, we will introduce the nested architecture to decompose multivariate dependence. After that, we will discuss our example of symbol detection in section III. The main contribution of this work is to develop a general method to design DNN modules that can utilize the available structural knowledge of the problem and, hence, can be integrated as a part of a large engineering system. The key concepts are the geometric information metrics and the DNN-based geometric operations to learn and approximate complex statistical dependence. We will summarize these ideas at the end of this paper.

II. Feature Space and Model Approximation

A. Feature Functions, Feature Space, and Joint Functions

For a random variable Z, we use $\mathcal{Z}$ to denote the corresponding alphabet, and use z to denote a specific value in $\mathcal{Z}$. We use P_{Z} to denote the probability distribution of Z, $\mathcal{P}^{\mathcal{Z}}$ to denote the collection of probability distributions supported on $\mathcal{Z}$, and $\mathrm{relint}(\mathcal{P}^{\mathcal{Z}})$ to denote the relative interior of $\mathcal{P}^{\mathcal{Z}}$. We refer to real-valued functions $f : \mathcal{Z} \to \mathbb{R}$ as *feature functions* on $\mathcal{Z}$, and we refer to *feature space* $\mathcal{F}_{\mathcal{Z}}$ as the collection of such feature functions. We define the inner product on $\mathcal{F}_{\mathcal{Z}}$ as

$$\langle f_1, f_2 \rangle \triangleq \mathbb{E}_R\left[f_1(\mathsf{Z}) f_2(\mathsf{Z})\right] \tag{1}$$

where $R \in \mathrm{relint}(\mathcal{P}^{\mathcal{Z}})$ is referred to as the *metric distribution*. To simplify the notation, once the metric distribution R is chosen, we shift every feature function to have $\mathbb{E}_R\left[f(\mathsf{Z})\right] = 0, \forall f$. With a little abuse of the notation, we still refer to the collection of such zero-mean feature functions $\mathcal{F}_{\mathcal{Z}}$. This defines $\mathcal{F}_{\mathcal{Z}}$ as a Hilbert space. The related geometric concepts will be frequently used in this paper, including the induced notions of subspace, rank, norm, projection, etc.[1]

In this paper, we use this geometric structure to describe the dependence between random variables. For a given joint distribution $P_{\mathsf{XY}} \in \mathrm{relint}(\mathcal{P}^{\mathcal{X}\times\mathcal{Y}})$, we consider the function $\mathsf{i}_{\mathsf{X};\mathsf{Y}} \in \mathcal{F}_{\mathcal{X}\times\mathcal{Y}}$ with

$$\mathsf{i}_{\mathsf{X};\mathsf{Y}}(x,y) = \frac{P_{\mathsf{X},\mathsf{Y}}(x,y)}{P_{\mathsf{X}}(x)P_{\mathsf{Y}}(y)} - 1, \quad \forall x, y \tag{2}$$

which is referred to as the *canonical dependence kernel* (CDK). We use the product distribution $R_{\mathsf{XY}} = P_{\mathsf{X}} \cdot P_{\mathsf{Y}}$ as the metric distribution for $\mathcal{F}_{\mathcal{X}\times\mathcal{Y}}$. The "$-1$" in the definition ensures that $\mathsf{i}_{\mathsf{X};\mathsf{Y}}$ is always centered, i.e., has zero-mean w.r.t. the metric distribution.

It is clear that there is a one-to-one correspondence between P_{XY} and $\mathsf{i}_{\mathsf{X};\mathsf{Y}}$. We refer to either one as a given "model" and try to describe the dependence between the two random variables defined by this model. In particular, we consider the two marginal function spaces, $\mathcal{F}_{\mathcal{X}}, \mathcal{F}_{\mathcal{Y}}$, with metric distributions $R_{\mathsf{X}} = P_{\mathsf{X}}, R_{\mathsf{Y}} = P_{\mathsf{Y}}$ resp. For given $f \in \mathcal{F}_{\mathcal{X}}$, $g \in \mathcal{F}_{\mathcal{Y}}$, we use $f \otimes g$ to denote their product $((x,y) \mapsto f(x) \cdot g(y)) \in \mathcal{F}_{\mathcal{X}\times\mathcal{Y}}$, and refer to such functions as *product functions*. Each product function $\gamma \in \mathcal{F}_{\mathcal{X}\times\mathcal{Y}}$ can be written as

$$\gamma = \sigma \cdot (f \otimes g), \tag{3}$$

where $\sigma = \|\gamma\| \geq 0$, and $f \in \mathcal{F}_{\mathcal{X}}, g \in \mathcal{F}_{\mathcal{Y}}$ satisfy $\|f\| = \|g\| = 1$. We refer to (3) as the *standard form* of product functions. More generally, for given k-dimensional features $f = [f_1, \ldots, f_k]^T \in \mathcal{F}_{\mathcal{X}}^k$ and $g = [g_1, \ldots, g_k]^T \in \mathcal{F}_{\mathcal{Y}}^k$, we use the tensor notation to write the joint function $f \otimes g \triangleq \sum_{i=1}^k f_i \otimes g_i$. When we evaluate the function value, we write $(f \otimes g)(x,y) = \sum_{i=1}^k f_i(x) \cdot g_i(y) = f(x) \cdot g(y)$, as the dot product of two vectors in $\mathbb{R}^k$.

B. Modal Decomposition and Low-rank Approximation

We start with defining the following operations on joint functions.

Definition 1 (Modal Decomposition)**.** *For a given $\mathcal{F}_{\mathcal{X}\times\mathcal{Y}}$, we define operator ζ on $\mathcal{F}_{\mathcal{X}\times\mathcal{Y}}$ as the optimal rank-1 approximation*

$$\zeta(\gamma) \triangleq \arg \min_{\substack{\gamma' :\, \gamma' = f\otimes g \\ f\in\mathcal{F}_{\mathcal{X}}, g\in\mathcal{F}_{\mathcal{Y}}}} \|\gamma - \gamma'\|^2, \quad \gamma \in \mathcal{F}_{\mathcal{X}\times\mathcal{Y}}. \tag{4}$$

In addition, for all $k \geq 1$, we define the operator ζ_k as $\zeta_1 \triangleq \zeta$, and

$$\zeta_k(\gamma) \triangleq \zeta\left(\gamma - \sum_{i=1}^{k-1} \zeta_i(\gamma)\right),$$

which we refer to as the k-th mode of γ. Then, we use $\zeta_{\leq k}(\gamma) \triangleq \sum_{i=1}^k \zeta_i(\gamma)$ and $r_k(\gamma) \triangleq \gamma - \zeta_{\leq k}(\gamma)$ to denote the superposition of the top k modes and the corresponding remainder, respectively.

For a given model $\mathsf{i}_{\mathsf{X};\mathsf{Y}}$, the decomposition above finds an approximate model as $\zeta_{\leq k}(\mathsf{i}_{\mathsf{X};\mathsf{Y}})$, which is the sum of the first k modes. This approximate form has some desired properties. We write in the standard form (see (3)) as

$$\zeta_{\leq k}(\mathsf{i}_{\mathsf{X};\mathsf{Y}}) = \sum_{i=1}^k \sigma_i \cdot (f_i^* \otimes g_i^*) \tag{5}$$

and the corresponding joint distribution as

$$\begin{aligned} \tilde{P}_{\mathsf{XY}}^{(k)} &= P_{\mathsf{X}}P_{\mathsf{Y}} \cdot (1 + \zeta_{\leq k}(\mathsf{i}_{\mathsf{X};\mathsf{Y}})) \\ &= P_{\mathsf{X}}P_{\mathsf{Y}} \cdot \left(1 + \sum_{i=1}^k \sigma_i \cdot (f_i^* \otimes g_i^*)\right) \end{aligned} \tag{6}$$

[1] In most cases, the choice of the metric distribution is clear from the context. Thus, we will not specify that from the notation, although one should note that the results might change if we choose a different metric distribution.

(5) can be viewed as a singular value decomposition (SVD) in the functional space, which has a desired orthogonality property that $\langle f_i^*, f_j^* \rangle = \langle g_i^*, g_j^* \rangle = \delta_{ij}$. Furthermore, it can be verified that

$$\mathbb{E}_{\tilde{P}_{\mathsf{XY}}^{(k)}} \left[f_i^*(\mathsf{X}) \cdot g_j^*(\mathsf{Y}) \right] = \sigma_i \cdot \delta_{ij}$$

That is, under the approximate model $\tilde{P}_{\mathsf{XY}}^{(k)}$, the dependence between X and Y can be fully specified as k parallel one-to-one correlation between feature pairs $f_i^*(\mathsf{X})$ and $g_i^*(\mathsf{Y}), i = 1, \ldots, k$. This is desirable for inference problems. For example, to estimate the value of $g_i^*(\mathsf{Y})$, the corresponding feature $f_i^*(\mathsf{X})$ is a sufficient statistic. Moreover, as the correlation coefficients σ_i's have a descending order, when we need to approximate the true model but are constrained by the number of features to use, it makes sense to take the first k modes as in (5).

It is worth mentioning that the standard form in (5) is chosen purely for convenience. In particular, the condition that feature functions are orthonormal is often unnecessary in practice. In this paper, we are interested in finding a low-rank approximation to a given model,

$$\min_{\substack{f_1,\ldots,f_k \in \mathcal{F}_{\mathcal{X}} \\ g_1,\ldots,g_k \in \mathcal{F}_{\mathcal{Y}}}} \left\| \mathsf{i}_{\mathsf{X};\mathsf{Y}} - \left(\sum_{i=1}^{k} f_i \otimes g_i \right) \right\|^2 \tag{7}$$

The optimizers of (7) are not unique. Any set of features $(f_1, \ldots, f_k)$ that span the same subspace as $(f_1^*, \ldots, f_k^*)$ in (5) has a corresponding feature set $(g_1, \ldots, g_k)$ that yield the same sum. We call the resulting model $\tilde{\mathsf{i}}_{\mathsf{X};\mathsf{Y}}^{(k)} = \sum_i f_i \otimes g_i$ the *rank-k approximation* of $\mathsf{i}_{\mathsf{X};\mathsf{Y}}$. We reserve the term *modal decomposition* only for the solution of (5) where the features are ordered and orthonormal.

C. The Local Approximation

In related works [2], [3], the modal decomposition problem is often considered with an extra "local assumption". While we do NOT require this assumption in the rest of this paper, it is worthwhile to briefly cover some of the ideas that are helpful to develop intuitions on our results.

Roughly speaking, the local assumption asserts that the probability distribution should be "close" to the metric distribution so that, for example, the density ratio function in (2) can be well approximated by $\log(P_{\mathsf{XY}}/P_{\mathsf{X}}P_{\mathsf{Y}}$, which is the point-wise mutual information and a common way to describe dependence. The local approximation means that the model P_{XY} is close to the product distribution $P_{\mathsf{X}}P_{\mathsf{Y}}$, i.e., the dependence between X and Y is weak.

Under this assumption, the geometric concepts we defined are well connected with some classical information theoretic measures and concepts. For example, it can be shown that the inner product (1) becomes the Fisher information; squared distance in $\mathcal{F}_{\mathcal{Z}}$ is approximately proportional to the K-L divergence, with the special case that $\frac{1}{2}\|\mathsf{i}_{\mathsf{X};\mathsf{Y}}\|^2 \approx I(\mathsf{X};\mathsf{Y})$; the collection of models with rank k (5) approximates a k-dimensional exponential family; and the model approximation problem (7) becomes a divergence minimization, or, an ML fitting problem.

Such connections can be conceptually useful when we interpret geometric operations in feature space with information-theoretic language. For example, a rank-k model in (6) would satisfy that $I(\mathsf{X};\mathsf{Y}) \approx \frac{1}{2}\|\mathsf{i}_{\mathsf{X};\mathsf{Y}}\|^2 = \frac{1}{2}\sum_{i=1}^{k}\sigma_i^2$. This can be read as decomposing the total dependence between X and Y as the sum of the strengths of individual modes. In a related way, one can also interpret the optimization problem (7) as choosing a set of k most "informative" features.

One can establish many other connections and correspondence with the local approximation assumption. There is extensive literature on this topic, which is surveyed in some of our earlier papers [2]. In this work, we will not expand on this topic. We will proceed without the local assumption in our development. We will, however, take for granted that the optimization (7) yields "desirable" features.

D. Learn Features From Data

The optimization (7) is a useful tool for feature learning. The optimal choice of k-dimensional features is considered "informative" and can be used instead of the high-dimensional raw data in inference tasks. In practice, we often cannot access the true model $\mathsf{i}_{\mathsf{X};\mathsf{Y}}$. In such cases, the empirical model of the training dataset is used instead.

Figure 1 shows a diagram of solving this optimization for a classification problem with a finite $\mathcal{Y}$, using the common deep neural network (DNN) settings. Here, we represent all the layers of the DNN up to the final classification layer as a module to select f. The final layer consists of links connecting the output of f to each class $\mathsf{Y} = y$, with weight $g(y)$ and bias $b(y)$. We denote the collection of these parameters as $g = [g(1), \ldots, g(|\mathcal{Y}|)] \in \mathbb{R}^{k \times |\mathcal{Y}|}$ and $\underline{b} = [b(1), \ldots, b(|\mathcal{Y}|)]^T \in \mathbb{R}^{|\mathcal{Y}|}$. This layer forms a linear combination of features $f(x) \cdot g(y) + b(y)$ for each y. Suppose we use a linear (identity) activation function; the output of the network is $\hat{\mathsf{i}}(x,y) \stackrel{\Delta}{=} f(x) \cdot g(y) + b(y)$ and is used as an approximation to the true model $\mathsf{i}_{\mathsf{X};\mathsf{Y}}$. From (2), this is equivalent as using

$$\tilde{P}_{\mathsf{Y}|\mathsf{X}}^{(f,g,b)}(y|x) = P_{\mathsf{Y}}(y) \cdot (1 + f(x) \cdot g(y) + b(y))$$

to approximate the true model $P_{\mathsf{Y}|\mathsf{X}}$. The choice of the bias values $b(\cdot)$ can be thought of as canceling the non-zero mean of $g(\mathsf{Y})$ w.r.t. P_{Y} so that the result is a valid distribution. Now if an appropriate L_2 loss function is used, we can readily train this network to find the f, g that solves (7). Some experimental results verifying the fact that DNN learning results coincide with the solution of (7) are reported in our previous works [2]. In practice, a non-linear activation function such as softmax can be used, and cross-entropy loss is often chosen instead of an L_2 loss. These can be viewed either as approximations or variations of (7). We will not expand on how to make these choices in this paper.

This observation, however, reveals a structural limitation of DNNs. The feature function of Y is represented as weights on individual links and only applies when Y takes discrete and

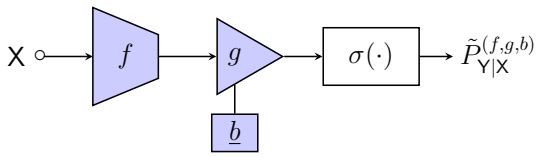

Figure 1: Deep neural network unit to compute low rank approximation (7).

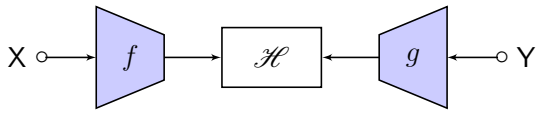

Figure 2: An H-Score Network: X and Y samples are processed separately by the f and g modules, which represent DNN modules chosen for the specific application; the outputs feature $f(\mathsf{X}), g(\mathsf{Y})$ are used to evaluate the H-score, which then used to adjust the DNN modules through backpropagation.

finite values. A more flexible approach is to use a concept called the H-score [4].

Definition 2 (H-Score)**.** *Given $k \geq 1$ and $f \in \mathcal{F}^k_{\mathcal{X}}$, $g \in \mathcal{F}^k_{\mathcal{Y}}$, the H-score $\mathscr{H}(f,g)$ is defined as*

$$\mathscr{H}(f,g) \triangleq \frac{1}{2}\left(\|\mathsf{i}_{\mathsf{X};\mathsf{Y}}\|^2 - \|\mathsf{i}_{\mathsf{X};\mathsf{Y}} - f\otimes g\|^2\right) \tag{8}$$

$$= \mathbb{E}\left[f^T(\mathsf{X})g(\mathsf{Y})\right] - \frac{1}{2}\cdot \operatorname{trace}\left(\Lambda_f \Lambda_g\right), \tag{9}$$

where $\Lambda_f \triangleq \mathbb{E}\left[f(\mathsf{X})f^T(\mathsf{X})\right]$ and $\Lambda_g \triangleq \mathbb{E}\left[g(\mathsf{Y})g^T(\mathsf{Y})\right]$, respectively.

By definition, when the model $\mathsf{i}_{\mathsf{X};\mathsf{Y}}$ is given, maximizing the H-score is equivalent to minimizing the approximation error in (7). There are some advantages of using the H-score maximization. First, from (9), the H-score can be computed from expectations. This can be naturally replaced by empirical averages in learning problems, where we cannot access the model but have a collection of samples. Figure 2 gives an illustration of this operation.

Secondly, the H-score directly measures the quality of features instead of borrowing the performance metrics of specific inference tasks. This means that the H-score maximization can operate even if the $\mathsf{X}-\mathsf{Y}$ dependence is weak and the reconstruction of the labels is difficult for DNNs. Moreover, the following property says that the value of the H-score has its information-theoretic upper bounds and the maximal value has direct operational meanings.

Property 3. *Given $k \geq 1$ and $f \in \mathcal{F}^k_{\mathcal{X}}$, $g \in \mathcal{F}^k_{\mathcal{Y}}$, we have*

$$\mathscr{H}(f,g) \leq \frac{1}{2}\left\|\zeta_{\leq k}(\mathsf{i}_{\mathsf{X};\mathsf{Y}})\right\|^2 = \frac{1}{2}\sum_{i=1}^{k}\sigma_i^2 \tag{10}$$

with $\sigma_i \triangleq \|\zeta_i(\mathsf{i}_{\mathsf{X};\mathsf{Y}})\|$ for $i = 1,\ldots,k$, where the inequality holds with equality if and only if $f\otimes g = \zeta_{\leq k}(\mathsf{i}_{\mathsf{X};\mathsf{Y}})$.

The most important advantage of the H-score maximization is that X and Y are processed separately without any restriction on the forms of these variables. This allows direct control of individual feature functions and flexible choices of DNN modules. It turns out that this fact is critical in applying DNN to engineering problems.

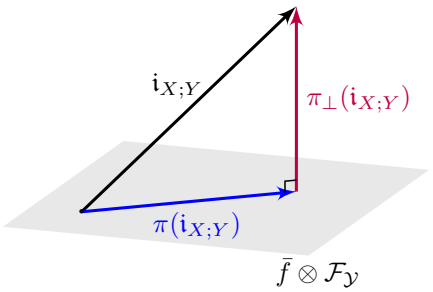

Figure 3: Orthogonal decomposition of the CDK function $\mathsf{i}_{X;Y}$.

E. Nested H-Score Networks

In this section, we consider a variation of (7),

$$\min_{f\in\mathcal{F}^k_{\mathcal{X}}, g\in\mathcal{F}^k_{\mathcal{Y}}: f\perp\bar{f}} \|\mathsf{i}_{\mathsf{X};\mathsf{Y}} - f\otimes g\|^2 \tag{11}$$

where $\bar{f}$ is a given feature function in $\mathcal{F}_{\mathcal{X}}$. [2]

The difference is that we would like to select features that are orthogonal to the given ϕ. There are many such examples that arise from applications where we have domain knowledge that restricts the search space of feature functions. There are also many options to work with the constraint. The most commonly used way is to include a regulator in the loss function. One can also choose the DNN module for selecting f to satisfy certain constraints. For example, CNN and RNN can be viewed as such specialized modules. Here, our goal is to use this example to demonstrate the flexibility of the H-score-based architecture. We will develop a nested architecture to perform the general projection operation in the feature space.

Our solution starts with changing Figure 2 by replacing the f module with the known feature function $\bar{f}$. We can still train the feature function for Y to maximize the H-score. This can be written as the optimization problem

$$\bar{g}^* = \arg\min_{\bar{g}\in\mathcal{F}_{\mathcal{Y}}} \left\|\mathsf{i}_{\mathsf{X};\mathsf{Y}} - \bar{f}\otimes\bar{g}\right\|^2 \tag{12}$$

This optimization searches for approximate model in $\{\gamma = \bar{f}\otimes\bar{g} : \bar{g}\in\mathcal{F}_{\mathcal{Y}}\}$, which is clearly a linear subspace of $\mathcal{F}_{\mathcal{X}\times\mathcal{Y}}$. We denote the resulting projection as $\pi(\mathsf{i}_{\mathsf{X};\mathsf{Y}}) = \bar{f}\otimes\bar{g}^*$, in Figure 3.

Now we observe that the approximation error, denoted as $\pi_\perp(\mathsf{i}_{\mathsf{X};\mathsf{Y}})$ in Figure 3, is orthogonal to the subspace. This implies that if we further find low-rank approximation to

[2]To simplify notation, we do not use different notations to distinguish a single feature function and a collection of feature functions since there is no conceptual difference in our development. We will continue to use the notation $f\otimes g = \sum_i f_i\otimes g_i$ without specifying the dimensionality of f, g. When $\bar{f}\in\mathcal{F}^m_{\mathcal{X}}$ for some $m > 1$, the orthogonality constraint means that every element of f is orthogonal to every $\bar{f}$, or $\operatorname{span}(f)\perp\operatorname{span}(\bar{f})$.

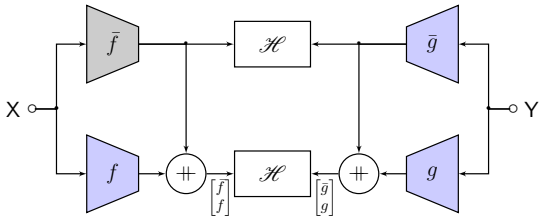

Figure 4: Nested H-Score Network for Projection of Feature Functions.

$\pi_\perp(\mathsf{i}_{\mathsf{X};\mathsf{Y}})$, all the features we find must be orthogonal to $\bar{f}$. Mathematically, the optimal solution of

$$f^*, g^* = \arg \min_{f \in \mathcal{F}_\mathcal{X}, g \in \mathcal{F}_\mathcal{Y}} \left\| \underbrace{\left(\mathsf{i}_{\mathsf{X};\mathsf{Y}} - \bar{f} \otimes \bar{g}^*\right)}_{\pi_\perp(\mathsf{i}_{\mathsf{X};\mathsf{Y}})} - f \otimes g \right\|^2 \quad (13)$$

$$= \arg \min_{f \in \mathcal{F}_\mathcal{X}, g \in \mathcal{F}_\mathcal{Y}} \left\| \mathsf{i}_{\mathsf{X};\mathsf{Y}} - (\bar{f} \otimes \bar{g}^* + f \otimes g) \right\|^2 \quad (14)$$

must satisfy that $f^* \perp \bar{f}$, and is the solution to the problem with constraint (11). Finally, we observe that in (14), the problem is equivalent to approximating $\mathsf{i}_{\mathsf{X};\mathsf{Y}}$ by two pairs of feature functions $(\bar{f}, \bar{g}^*)$ and (f, g), which gives rise to the nested H-score network shown in Figure 4.

The architecture in Figure 4 can be understood as the following. First, we train the upper branch, with the $\bar{f}$ module frozen as the given constraint function. This allows the $\bar{g}$ module to optimize at $\bar{g}^*$ defined in (12). The symbol "$+\!\!+$" denotes "concatenation" to form features $\begin{bmatrix} \bar{f} \\ f \end{bmatrix}$ and $\begin{bmatrix} \bar{g} \\ g \end{bmatrix}$, which are optimized through the lower branch to solve (14). In practice, we do not wait for the upper branch optimization to converge but run the entire network together, which requires computing the sum of the two H-scores in each iteration. When this process converges, we get the desired optimal feature f^* that satisfies the orthogonality constraint.

Compared to the other existing methods to solve the constrained optimization problem (11), the nest H-score method is based on a systematic approach to make projections in the feature space. In the next section, we will use an example to illustrate that this concept can be generalized to learning problems with more complex dependence structures.

III. Case Study: Receiver with Side Information

A. The Symbol Detection Problem

In this section, we consider a problem of symbol detection in fading channels with interference as follows.

$$\mathsf{Y} = h_1 \cdot \mathsf{X}_1 + h_2 \cdot \mathsf{X}_2 + \mathsf{W} \quad (15)$$

where $\mathsf{X}_1 \in \{+1, -1\}$ is the BPSK transmitted signal that we wish to detect; X_2 is the symbol transmitted by an interferer, taking value equally likely from the 16-QAM constellation; W is the additive Gaussian noise with distribution $\mathcal{CN}(0, \sigma_W^2)$,

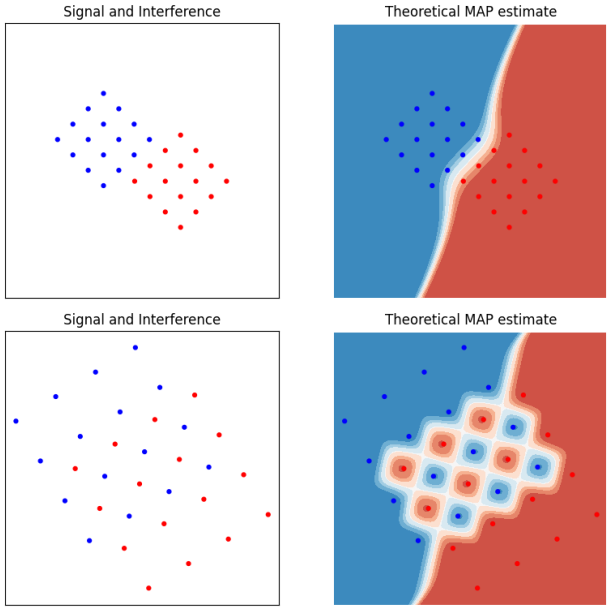

Figure 5: Two cases of the optimal decision functions: the sum of the transmitted signal and the interference $h_1\mathsf{X}_1 + h_2\mathsf{X}_2$, without the additive noise, are shown on the left; the red and blue color represent the two hypotheses we need to decide on; the received signal follows the mixture Gaussian distribution with these points as centers. The right side shows the color-coded contour of the decision function.

with the variance inversely proportional to the average signal-to-noise ratio (SNR); $h_1, h_2 \in \mathbb{C}$ are the fading coefficients for the target transmitter and the interferer, resp.; and Y is the received signal. We assume the channel state information (CSI), $\mathsf{S} = (h_1, h_2, \mathsf{SNR})$, follows a known distribution P_S. The realization $\mathsf{S} = s$ is known at the receiver. Our goal is to design a receiver that can decide the value of X_1 based on the received signal.

This is a well-studied problem in communications. The optimal MAP decision rule evaluates the sign of the likelihood ratio $\log\left(P_{\mathsf{Y}|(\mathsf{X}_1=+1),\mathsf{S}}(y|s)/P_{\mathsf{Y}|(\mathsf{X}_1=-1),\mathsf{S}}(y|s)\right)$. Each likelihood is a mixture Gaussian distribution

$$P_{\mathsf{Y}|\mathsf{X}_1,\mathsf{S}}(y|x_1, s) = \sum_{x_2 \in 16\text{QAM}} \frac{1}{16} \cdot \mathcal{CN}(y; h_1 x_1 + h_2 x_2, \sigma_W^2)$$

where $\mathcal{CN}(\cdot; \mu, \sigma^2)$ denotes the complex Gaussian density with the given mean and variance. Figure 5 shows two examples of this decision function.

While the optimal receiver is not difficult to derive analytically, implementing such functions in practice can be cumbersome because of the non-linearity of these functions and because the optimal decision is rather sensitive to parameter changes. DNNs are a desirable implementation option since, in principle, they can universally approximate all non-linear functions with standardized hardware and software packages. It is indeed not difficult to train a neural network to form near-optimal decisions, even for the highly non-linear case in

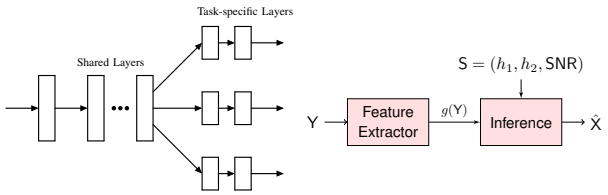

Figure 6: (a) Multi-task Learning; (b) Learning with Side-information

the figure. The real challenge in this problem is that we need not just the optimal decision maker for an isolated scenario, but rather a parameterized optimal solution. That is, when we change the system parameters, the CSI, at the receiver, we want to have the optimal receiver for all cases corresponding to the different CSI values. It is unclear what data we should use to train the neural network, and how to control the behavior of the neural network with the given parameters. The same difficulty exists in a wide variety of engineering problems. We will use this problem as an example to illustrate how to solve such problems with the nested H-score network proposed in the previous section.

In the learning literature, the problem is related to a multi-task learning setting [5], [6], shown in Figure 6-(a). The neural network is organized as several shared layers followed by some task-specific layers. The shared layers can be trained offline, often with simulated samples drawn from a mixture of all fading environments. The task-specific layers are used to tune the decoder to work with the specific fading environment, i.e. for the given CSI value. Consequently, these task-specific layers must be trained with online samples drawn from the case of interest. In practice, the online samples are expensive since they are the transmitted and the received signals over the wireless channel. This creates a tension that limits the adaptability of the overall system. Particularly since we operate DNNs as black boxes and do not have a way to measure how well the decoder can adapt to the specific fading scenario, the performance of such solutions can be hard to predict.

The classical information-theoretic treatment of the problem is as a receiver with side-information problem shown in Figure 6(b). [3] One can intuitively think the processing of the received signal Y as extracting useful information to make decisions on $\hat{\mathsf{X}}$ in presence of the side-information S. This relation is represented as the chain rule of information:

$$I(\mathsf{Y};(\mathsf{S},\mathsf{X})) = I(\mathsf{Y};\mathsf{S}) + I(\mathsf{Y};\mathsf{X}|\mathsf{S}) \tag{16}$$

where we can only hope to decode the information from the second conditional term. In the context of coded transmissions, the chain rule has clear operational meanings associated with joint typically decoding and random binning techniques. Here, our goal is to find the operations corresponding to the chain rule in the context of learning the model of $\mathsf{X},\mathsf{S},\mathsf{Y}$, where we would like to find informative features that can be used to make inferences with the given side information.

[3]Different from the previous notation, we start at this point to use Y, the received signal, as the input to our system.

B. Decomposition of Multivariate Dependence

Using the concepts of modal decomposition developed in Section II, we would like to learn features to approximate the dependence between the received signal Y and the pair of random variables S,X. We consider the model CDK $\mathsf{i}_{\mathsf{Y};(\mathsf{S},\mathsf{X})} \in \mathcal{F}_{\mathcal{Y}\times\mathcal{S}\times\mathcal{X}}$. We choose the metric distribution as $R = P_{\mathsf{Y}} \cdot P_{\mathsf{SX}}$. In order to separate the two parts of information in the chain rule (16), we consider the following optimization problem

$$\pi_{\mathrm{M}}(\mathsf{i}_{\mathsf{Y};(\mathsf{S},\mathsf{Y})}) \triangleq \arg \min_{\substack{\hat{\mathsf{i}}\in\mathcal{F}_{\mathcal{Y}\times\mathcal{S}\times\mathcal{X}}:\\ \mathsf{Y}-\mathsf{S}-\mathsf{X}}} \|\mathsf{i}_{\mathsf{Y};(\mathsf{S},\mathsf{Y})} - \hat{\mathsf{i}}\|^2 \tag{17}$$

where the constraint is that the approximate model $\hat{\mathsf{i}}$ corresponds to a joint distribution $\hat{P}_{\mathsf{YSX}}$ that satisfies the Markov relation $\mathsf{Y}-\mathsf{S}-\mathsf{X}$. That is, under $\hat{P}_{\mathsf{YSX}}$, Y depends on the pair (S,X) only through S alone.

Next, we observe that the collection of models $\hat{\mathsf{i}}$ satisfying the Markov relation forms a linear subspace in $\mathcal{F}_{\mathcal{Y}\times\mathcal{S}\times\mathcal{X}}$, since the function $\hat{\mathsf{i}}(y,s,x)$ for any y,s,x does not depend on x, which is a linear constraint. (17) can thus be viewed as a projection to a linear subspace, depicted in Figure 7. We thus can understand the projection $\pi_{\mathrm{M}}(\mathsf{i}_{\mathsf{Y};(\mathsf{S},\mathsf{X})})$ as the "Markov component" of the given model. Following the intuition of the chain rule (16), we denote the approximation error as $\pi_{\mathrm{C}}(\mathsf{i}_{\mathsf{Y};(\mathsf{S},\mathsf{X})})$, which is understood as the "conditional dependence" component of the given model.

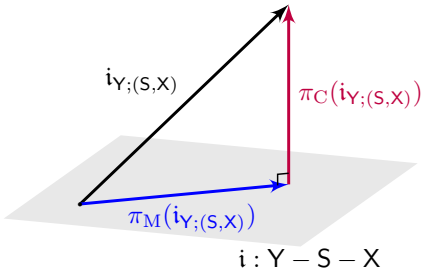

Figure 7: Decomposition of Multivariate Dependence

If we are willing to take the extra assumption that the dependence between Y and (S,X) is weak, i.e., the model P_{YSX} is close to the product distribution $R = P_{\mathsf{Y}} \cdot P_{\mathsf{SX}}$, which we take as the metric distribution, it can be further established that $I(\mathsf{Y};\mathsf{S}) \approx \frac{1}{2}\|\pi_{\mathrm{M}}(\mathsf{i}_{\mathsf{Y};(\mathsf{S},\mathsf{X})})\|^2$, and $I(\mathsf{Y};\mathsf{X}|\mathsf{S}) \approx \frac{1}{2}\|\pi_{\mathrm{C}}(\mathsf{i}_{\mathsf{Y};(\mathsf{S},\mathsf{X})})\|$. We thus can view the optimization (17) and Figure 7 as the operational meaning of the chain rule (16) in the functional space $\mathcal{F}_{\mathsf{Y}\times\mathsf{S}\times\mathsf{X}}$: the two parts of the overall dependence are separated into two orthogonal subspaces.

This operation is indeed similar to what we discussed in section II-E, which can be learned with a nested-H-score network directly from data samples, as shown in Figure 8. In the figure, the upper link learns a pair of functions $\bar{g} \in \mathcal{F}_{\mathcal{Y}}$, $\bar{f} \in \mathcal{F}_{\mathcal{S}}$, such that $\bar{g} \otimes \bar{f}$ is a good approximation of the Markov component $\pi_{\mathrm{M}}(\mathsf{i}_{\mathsf{Y};(\mathsf{S},\mathsf{X})})$, which captures the dependence between Y and

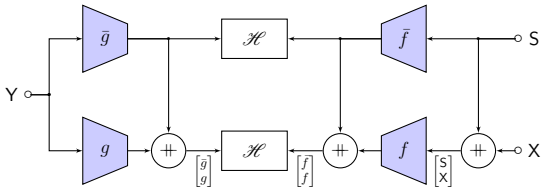

Figure 8: Nesting architecture for decomposition of multivariate dependence

S. The lower link learns $g \in \mathcal{F}_{\mathcal{Y}}, f \in \mathcal{F}_{\mathcal{S}\times\mathcal{X}}$ so that $g \otimes f$ approximates the conditional element $\pi_{\mathrm{C}}(\mathsf{i}_{\mathsf{Y};(\mathsf{S},\mathsf{X})})$. This captures the information in Y complementary to the part carried by $\bar{g}$, i.e., the part that cannot be inferred from the side information S alone.

C. Feature Assembling and Inference Models

The nested H-score network in Figure 8 learns feature functions $\bar{f}, \bar{g}, f, g$ to have the following approximations:

$$\begin{aligned} P_{\mathsf{SY}}(s,y) &\approx P_{\mathsf{S}}(s)\cdot P_{\mathsf{Y}}(y)\cdot(1+\bar{f}(s)\cdot\bar{g}(y)) \\ P_{\mathsf{XSY}}(x,s,y) &\approx P_{\mathsf{X}}(x)P_{\mathsf{S}}(s)\cdot P_{\mathsf{Y}}(y) \\ &\quad\cdot(1+\bar{f}(s)\cdot\bar{g}(y)+f(s,x)\cdot g(y)) \end{aligned} \tag{18}$$

where we used the fact that S, the CSI, and X, the transmitted symbol, are independent in this symbol detection problem. Maximizing the H-scores in this network is equivalent to minimizing the L_2 distances in these approximations. Once the training process converges, we can assemble the resulting neural network modules to make our desired detector. In this case, we hope to predict the value of X from the observation of Y and the side information S, thus we use (18) to form an approximation

$$\hat{P}_{\mathsf{X}|\mathsf{YS}}(x|y,s) \triangleq P_{\mathsf{X}}(x)\cdot\left(\frac{1+\bar{f}(s)\cdot\bar{g}(y)+f(x,s)\cdot g(y)}{1+\bar{f}(s)\cdot\bar{g}(y)}\right) \tag{19}$$

This can be evaluated with the observed value of y, s and all possible values of x (binary in this case), and the maximizer is chosen as the decision.

Remark. *We did not specify the choices of all four neural network modules, which impose constraints on the dimensionality and the expressive power and dictate how well the approximations are. One major advantage of this method is that the training process can be carried out offline with simulated data samples. There is no need to use any online samples to adapt the receiver to the specific fading environment. Our knowledge of the fading environment is summarized in the form of estimated channel parameters and directly entered into the solution as inputs to feature functions. This alleviates the tension of using expensive online samples in practice and allows us to choose more powerful neural network modules with more extensive training. The result is what we usually have with model-based methods: a parameterized solution that performs well in all situations whenever we give it the right value of the parameters.*

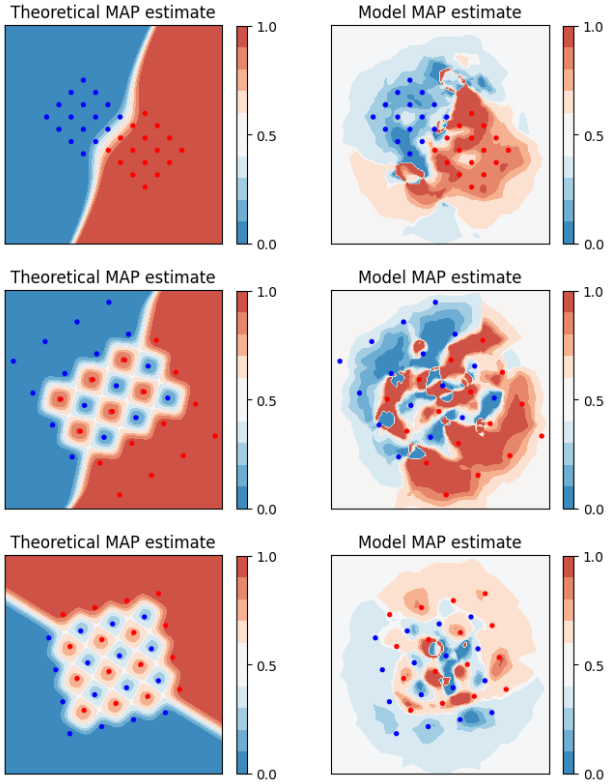

Figure 9: The decision functions in a few cases: the learned function varies with the channel parameters.

The specific way to decompose the dependence between Y and the pair of random variables (S,X) into the Markov and the conditional components is not critical in our design. Other ways to approximate the joint distribution by feature functions might also work. For example, one could directly approximate the model $\mathsf{i}_{\mathsf{Y};(\mathsf{S},\mathsf{X})}$ with $g(\mathsf{Y})\cdot f(\mathsf{S},\mathsf{X})$, without separating the two components; and use Bayes' rule to assemble the learned features to a decision maker.

Our decomposition has the additional advantage that $g(\mathsf{Y})$ does not carry information about the side information S. If we have a distributed processing problem where we need to restrict the information volume of the output of the front-end processing of the received signal Y, or if we would like to avoid leaking information about S for some security reason, the choice of feature function $g(\cdot)$ in our design would be ideal. In practice, it is observed that such decomposed neural network modules are easier to train compared to directly learning the overall multivariate model. Conceptually, it is valuable to separate the contribution of different variables in a multivariate dependence.

Figure 9 shows the learned decision function at a few different CSI values. The left column shows the theoretical optimal decision function in these cases in comparison to the decision function assembled with the neural network modules following (19). One can observe that the centers of the Gaussian mixtures are all decided correctly (with the

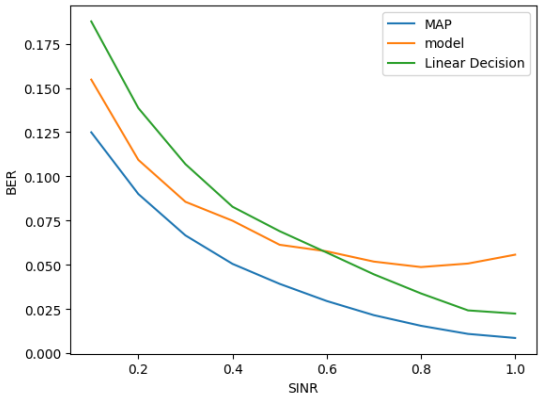

Figure 10: The BER performance of the learned decision maker.

correct color). More importantly, the learned decision function can follow the theoretical optimal as the CSI value changes. This is achieved without any retraining with online samples. The information about the specific realization of the fading environment is provided only with the CSI value.

As expected, being able to use non-linear functions in the receivers correctly helps to improve the performance, particularly when the interference is strong. Figure 10 shows the bit error rate achieved with our design. When the SINR is low, the model using our approach has a significant advantage over the linear receivers, i.e. threshold after the optimal linear processing.

IV. Concluding Remarks

In this paper, we reported some of our latest practice of using neural networks as a component of an engineering system to utilize the computation power and the capability of non-linear processing to improve the overall system performance. We believe such goals are shared by a wide range of current research efforts in different engineering areas. There are also some shared difficulties in these practices, mainly due to the fact that DNNs are usually used as black boxes, giving us insufficient control and performance metrics to fully integrate DNNs into the existing engineering solutions. We hope that the study case of symbol detection in this paper can shed some light on overcoming this difficulty in more general contexts.

Specifically, we found the following general principles are helpful in adopting DNNs in engineering designs.

First, to "open the black box", we should not always train a neural network as a classifier and try to use it as a classifier. In our approach, neural network modules are trained to learn certain feature functions. f, g in Figure 2 and $\bar{f}, \bar{g}, f, g$ in Figure 8. These feature functions are well-defined from the corresponding optimization problems, and thus, each comes with clear operational meanings. We then have a separate "assembling" process to put these learned feature functions together for a specific inference task, such as in (19). In the more basic setting (7), we can use the learned f, g features to either estimate the value of Y with the input X, or reversely estimate X from Y. This separation between the learning process and the use of the learning results at the level of feature functions provides the critical flexibility that we need in engineering problems.

Second, having quality metrics for features is instrumental to operating on feature functions. Current practices using DNNs often "borrow" task-specific performance metrics to encourage the neural networks to learn informative features. If we can correctly predict the value of Y with a neural network, then the internal processing of this neural network must take the right information. This logic is not wrong but very inflexible, making it hard to move a neural network that works well for one task to work for another. Conceptually, even if X and Y are weakly dependent, such that we cannot decide, with a desirable accuracy, the value of Y from the value of X, the strongest dependence mode between the two is still well-defined and should be accessible when we have enough samples. The H-score is one way to directly define a quality metric of feature functions, which can be evaluated with data samples. Conceptually, it allows us to solve the low-rank approximation problem (7) regardless of how close the resulting approximation can be.

Third, the operations on feature functions are, by nature, high-dimensional geometric operations. The key concepts and operations we discussed in this paper all have geometric meanings on $\mathcal{F}_{\mathcal{Z}}$, such as inner product, projections, subspaces, etc. We believe this is the key reason that when we use scalar-valued metrics to describe the operations of neural networks, either a task-specific loss function or an information-theoretic quantity like mutual information, it is often hard to get the complete picture. This effect is more prominent in multivariate problems where the statistical dependence gets more complex. We thus consider developing geometry-based analysis tools the right move for understanding these problems.

References

[1] W. Yu, F. Sohrabi, and T. Jiang, "Role of deep learning in wireless communications," *IEEE BITS the Information Theory Magazine*, vol. 2, no. 2, pp. 56–72, 2022.

[2] S.-L. Huang, X. Xu, and L. Zheng, "An information-theoretic approach to unsupervised feature selection for high-dimensional data," *IEEE Journal on Selected Areas in Information Theory*, vol. 1, no. 1, pp. 157–166, 2020.

[3] X. Xu and L. Zheng, "A geometric framework for neural feature learning," *arXiv:2309.10140*, 2023.

[4] X. Xu, S.-L. Huang, L. Zheng, and G. W. Wornell, "An information theoretic interpretation to deep neural networks," *Entropy*, vol. 24, no. 1, p. 135, 2022.

[5] R. Caruana, "Multitask learning: A knowledge-based source of inductive bias1," in *Proceedings of the Tenth International Conference on Machine Learning*, pp. 41–48, Citeseer, 1993.

[6] S. Ruder, "An overview of multi-task learning in deep neural networks," *arXiv preprint arXiv:1706.05098*, 2017.