machine learning & knowledge extraction

MDPI

Article

Multi-Task Representation Learning for Renewable-Power Forecasting: A Comparative Analysis of Unified Autoencoder Variants and Task-Embedding Dimensions

Chandana Priya Nivarthi *, Stephan Vogt and Bernhard Sick

Intelligent Embedded Systems, University of Kassel, Wilhelmshöher Allee 73, 34121 Kassel, Germany; stephan.vogt@sma.de (S.V.); bsick@uni-kassel.de (B.S.)
* Correspondence: chandana.nivarthi@uni-kassel.de

Abstract: Typically, renewable-power-generation forecasting using machine learning involves creating separate models for each photovoltaic or wind park, known as single-task learning models. However, transfer learning has gained popularity in recent years, as it allows for the transfer of knowledge from source parks to target parks. Nevertheless, determining the most similar source park(s) for transfer learning can be challenging, particularly when the target park has limited or no historical data samples. To address this issue, we propose a multi-task learning architecture that employs a Unified Autoencoder (UAE) to initially learn a common representation of input weather features among tasks and then utilizes a Task-Embedding layer in a Neural Network (TENN) to learn task-specific information. This proposed UAE-TENN architecture can be easily extended to new parks with or without historical data. We evaluate the performance of our proposed architecture and compare it to single-task learning models on six photovoltaic and wind farm datasets consisting of a total of 529 parks. Our results show that the UAE-TENN architecture significantly improves power-forecasting performance by 10 to 19% for photovoltaic parks and 5 to 15% for wind parks compared to baseline models. We also demonstrate that UAE-TENN improves forecast accuracy for a new park by 19% for photovoltaic parks, even in a zero-shot learning scenario where there is no historical data. Additionally, we propose variants of the Unified Autoencoder with convolutional and LSTM layers, compare their performance, and provide a comparison among architectures with different numbers of task-embedding dimensions. Finally, we demonstrate the utility of trained task embeddings for interpretation and visualization purposes.

Keywords: transfer learning; multi-task learning; zero-shot learning; autoencoders; power forecast

Citation: Nivarthi, C.P.; Vogt, S.; Sick, B. Multi-Task Representation Learning for Renewable-Power Forecasting: A Comparative Analysis of Unified Autoencoder Variants and Task-Embedding Dimensions. *Mach. Learn. Knowl. Extr.* **2023**, *5*, 1214–1233. https://doi.org/10.3390/make5030062

Academic Editors: Moamar Sayed-Mouchaweh, Mohd Arif Wani, Vasile Palade and Mehmed M. Kantardzic

Received: 19 July 2023
Revised: 4 September 2023
Accepted: 8 September 2023
Published: 20 September 2023

1. Introduction

The escalating rise in global surface temperatures emphasizes the urgent need to shift rapidly towards renewable-energy sources for generating power. As outlined in the 2021 global roadmap for Sustainable Development Goal (SDG) 7, which centers on attaining accessible and environmentally friendly energy, in conjunction with the stipulations of the Paris Agreement on climate change, there is a pressing appeal to triple the worldwide potential for renewable energy by 2030 and to realize a state of net-zero emissions by 2050 [1]. Meeting this capacity increase also requires improved accuracy in predicting renewable-power generation. Accurate power forecasts are crucial for lowering operational expenses and enhancing the safety and maintenance of power grids.

In the context of renewable energy, typical day-ahead power forecast models rely on Numerical Weather Prediction (NWP) data for the upcoming 24 to 49 h. Weather factors like sunlight and wind speed are vital in estimating power generation for solar and wind parks, respectively. Various models, including physical, statistical, and machine learning techniques, have been developed to enhance forecasting accuracy. Deep learning methods have recently gained attention for power forecasting [2]. Most of these models aim to

improve forecasting accuracy for individual solar or wind parks by learning the unique traits of each park. It involves creating a separate model for each park, called single-task learning (STL). However, as the number of solar and wind parks grows, developing and training numerous models becomes resource-intensive. This approach can inadvertently lead to a significant carbon footprint [3], which contradicts the purpose of renewable-energy systems.

To address this challenge, Transfer Learning (TL) and Multi-task Learning (MTL) offer potential solutions by leveraging the relationships between renewable-energy parks to share knowledge and reduce environmental impact. TL can improve forecasts through retraining, particularly when a target park has enough historical data [4]. However, when historical data are limited, MTL becomes helpful in making predictions. MTL involves learning multiple tasks simultaneously to benefit from insights between tasks [5]. MTL comprises two key aspects: learning standard information independent of tasksand learning specific information for each task. Autoencoders (AE) effectively capture shared representations of tasks [6]. Task embeddings, used to encode task-specific information, are also gaining traction in MTL.

This paper proposes a novel MTL approach that combines the Unified Autoencoder (UAE) concept with Task Embedding in a Neural Network (TENN) architecture. The UAE approach involves training an autoencoder on consolidated data from all tasks to capture shared, task-independent information. The TENN architecture uses an embedding layer in a neural network to encode task identity information. We will compare the proposed Unified Autoencoder-Task-Embedding Neural Network (UAE-TENN) with various STL and MTL methods. Notably, we are interested in assessing the UAE-TENN's performance in the scenario of power forecasting for a new park, where historical data are absent, known as zero-shot learning (ZSL). This evaluation is pivotal for reliable power forecasts in new parks without historical data, aiding operational planning. This study seeks to address the following research questions:

Question 1: Does unified encoding of task-independent features along with task embeddings decrease the forecast error for PV and wind parks compared to STL methods?
Question 2: Can autoencoder-based MTL architecture with TE perform better than using either UAE or TE in MTL architecture?
Question 3: Are autoencoder-based MTL architectures with TE capable of providing better forecasts in the ZSL scenario for renewable-power forecasts compared to the baseline model?
Question 4: Can convolutional and LSTM autoencoders improve the performance compared to regular fully connected layers for STL power forecast of photovoltaic datasets?
Question 5: Do convolutional and LSTM autoencoders do a better power forecast in a zero-shot learning scenario compared to a unified regular autoencoder in the proposed architecture?
Question 6: Does an increase in the number of task-embedding dimensions in the proposed architecture improve the performance of different autoencoder models?

The assessment of the proposed UAE-TENN architecture through experimentation provides the following significant contributions:

- Introducing a simplified MTL architecture, UAE-TENN, which demonstrates notably superior performance compared to STL and MTL baseline models.
- Contributing to the early stages of research in the field of ZSL for renewable-energy systems by showcasing the substantial advantage of UAE-TENN over baseline models in ZSL scenarios.
- Conducting an extensive experimental evaluation involving six datasets related to photovoltaic (PV) and wind sources, collectively encompassing 529 parks.
- Performing an ablation study on the UAE and TE components, indicating that the removal of either of these components results in decreased performance. Notably, the TE component has a more pronounced impact on performance enhancement in UAE-TENN than the UAE component.

- Comparing Unified Autoencoder architectures that incorporate convolutional and LSTM layers for this specific application. Evaluating the influence of task-embedding dimensionality on the performance of the proposed architecture.
- Demonstrating the utility of task embeddings for purposes of interpretation and visualization.
- Highlighting the ease with which the proposed architecture can be extended to various other domains.

In summary, this article contributes by introducing a simplified MTL architecture for renewable-power forecasting, highlighting early ZSL research, conducting extensive experimental evaluations among competitive models, and demonstrating the architecture's applicability across broader domains. The remainder of this article is structured as follows. Section 2 describes related work and Section 3 introduces the proposed method. The data sets, experimental evaluation, and findings are described in Sections 4–6. The utility of task embeddings for visualization is described in Section 7 followed by conclusion and outlook for future research in Section 8.

2. Related Work

To address our research questions, we review the related work encompassing single-task learning (STL), multi-task learning (MTL), and zero-shot learning (ZSL), with a specific focus on predicting renewable-energy power. Typically, predicting renewable power involves using past Numerical Weather Prediction (NWP) data containing weather-related attributes as input, while the corresponding power-generation data are considered the target. Comprehensive surveys such as [7] for Transfer Learning (TL) and [5] for Multi-Task Learning (MTL) provide insights into their general concepts. A recent overview [2] highlights the growing prevalence of deep learning architectures in renewable-power prediction and importantly notes the underutilization of autoencoders (AEs) for the same purpose.

However, AEs have found widespread utilization in studies focusing on images and text data, especially for tasks involving classification. The adaptability of AEs in acquiring versatile representations has been highlighted across diverse applications [8]. The authors of [9–11] suggested AEs for applications involving Transfer Learning (TL) and Multi-Task Learning (MTL) using image datasets for classification tasks. Furthermore, ref. [12] employs AEs to learn sentence representations in unsupervised texts. Notably, AEs have received substantial attention across various domains; however, their utilization in regression tasks, both in general and specifically within the domain of renewable-energy systems, has remained relatively limited. Among the initial instances of AE application in the renewable-energy sector, a study employed transfer learning by training multiple AEs on a single wind park, subsequently adapting them to other wind parks. This approach was augmented by a deep belief network that amalgamated features extracted from individual parks [13]. Similarly, a limited number of studies [14,15] proposed AEs for inductive transfer learning, focusing on wind power forecasting. In [14], a Unified Autoencoder was trained on wind turbine data to derive latent features, subsequently fine-tuned for final power prediction. Within the context of renewable-power forecasting, a few initial studies conducted a comparative assessment of diverse deep learning algorithms within a single-task learning (STL) framework. AEs were incorporated into a hybrid model, involving the initial training of a foundational autoencoder using NWP inputs, followed by further training of a neural network connected to the encoder for power forecasting. Similarly, ref. [16] introduced a stacked autoencoder for wind speed and power prediction. This pre-trained model was subsequently fine-tuned for target task experiments conducted at different locations. An alternate variant of the stacked autoencoder was proposed in [17] for wind power prediction. In this approach, the autoencoder was co-trained with the power prediction task, encompassing multi-output predictions for varying horizons. Notably, while these studies examined AE applications, the exploration of MTL-based autoencoders and their relevance in ZSL contexts was not explored. Furthermore, none of these studies leveraged a unified AE for representation learning within MTL tasks.

In the context of MTL, the commonly adopted architecture involves Hard Parameter Sharing (HPS), where each task is allocated one or more task-specific final layers within the model. However, extending this HPS approach to accommodate new parks that lack close relations to existing ones is challenging [18]. Embeddings have gained significant popularity in representing categorical entities like words in language models [19], and their broader applicability to different entity categories is demonstrated in [20]. Employing such embeddings as task embeddings (TE) to capture task-specific features has been explored in [4] within the domain of Transfer Learning (TL). Although AEs and TE have been independently investigated in the context of renewable-energy applications, this study endeavors to underscore the effectiveness of their combination. Additionally, limited research exists on ZSL in renewable-energy applications. Furthermore, to the best of our knowledge, no prior study has undertaken a comparative analysis of STL and MTL methods across six datasets, encompassing a total of 529 PV and wind parks.

3. Proposed Method

In this section, we introduce the relevant definitions of STL, MTL, and ZSL methods and then describe our proposed architecture.

3.1. Definitions of STL, MTL and ZSL

The general definition of MTL has been introduced in [5] and we adapted it to STL, ZSL for consistent formulation, similar to the terminology in [4].

A domain is defined by $\mathcal{D} = \{\mathcal{X}, P(X)\}$, where $\mathcal{X}$ is the feature space and $P(X)$ is the corresponding marginal distribution with $X = \{\mathbf{x} \mid \mathbf{x}_i \in \mathcal{X}, i = 1, \ldots, N\}$. Given a specific domain, a task consists of a label space $\mathcal{Y}$ and a conditional probability distribution $P(Y|X)$ and is represented as $\mathcal{T} = \{\mathcal{Y}, P(Y|X)\}$, where $P(Y|X)$ is learned from the training instances $(\mathbf{x}_i, y_i)$ with $\mathbf{x}_i \in \mathcal{X}$ and $y_i \in \mathcal{Y}$, where $Y = \{y \mid y_i \in \mathcal{Y}, i = 1, \ldots, N\}$. Here, $\mathbf{x}_i$ denotes the NWP data as input features, and y_i denotes the historical power measurements of a park as the target feature.

Definition 1. ***Single-task Learning*** *Given M tasks, when each task $\mathcal{T}$ is learned separately on the specific training data $(\mathbf{x}_i^m, y_i^m)$, where $m \in \{1, \ldots, M\}$, $i \in \{1, \ldots, N\}$ and $M \in \mathbb{N}^+$ tasks, it is termed as single-task learning.*

Definition 2. ***Multi-task Learning*** *Given M tasks, when all tasks or a subset of them are related, MTL aims to learn all M tasks together improving the performance of each task, using the knowledge gained from other tasks. In MTL approaches, the domain $\mathcal{D}_m$ of each task has training instances $(\mathbf{x}_i^m, y_i^m)$, $i \in \{1, \ldots, N\}$ where $\mathbf{x}_i^m \in X^m$, $y_i^m \in Y^m$, $m \in \{1, \ldots, M\}$, and $M \in \mathbb{N}^+$ tasks. Here, typically a single model is built using training data from multiple tasks.*

Definition 3. ***Zero-Shot Learning*** *Given a new task, which does not have any training data instances $(\mathbf{x}_i^m, y_i^m)$, the metadata of such a task is used to select the prediction function $f(\cdot)$ from the already trained similar tasks models to do a prediction.*

In contrast to the typical inductive TL approach, where the source task model is fine-tuned for the target task on target data, the ZSL task does not have any input or output data at the target task, making the problem more challenging.

3.2. Unified Autoencoder—Task-Embedding Neural Network

In Multi-Task Learning (MTL), neural network architectures, two predominant approaches are typically employed: Hard Parameter Sharing (HPS) and Soft Parameter Sharing (SPS) of hidden layers [18]. HPS architectures incorporate shared task-independent layers for all tasks alongside dedicated task-specific layers. On the other hand, SPS approaches introduce constraints to minimize the differences between parameters associated with different tasks, rather than directly sharing parameters. Both these popular approaches

share a fundamental principle: the acquisition of common features across all tasks and the capture of task-specific features to facilitate MTL. Following the same principle, we introduce an architecture that combines a Unified Autoencoder (UAE) with a neural network featuring a Task Embedding (TE) layer, for MTL in renewable-power forecasting. The UAE undertakes the learning of a shared, task-independent representation, while the TE layer embedded within the neural network captures task-specific insights. This section provides an overview of these two architectural components, followed by a discussion of the training and inference procedures.

3.2.1. Unified Autoencoder (UAE)

Autoencoders aim to learn the latent representation of inputs by reconstructing them. To fulfill this purpose, we employ a Unified Autoencoder (UAE) to learn latent representations of input features extracted from all photovoltaic (PV) or wind parks within a given dataset, as depicted in Figure 1. In the context of MTL, an AE unifies the feature representations across all training tasks, leading to its name as Unified Autoencoder (UAE).

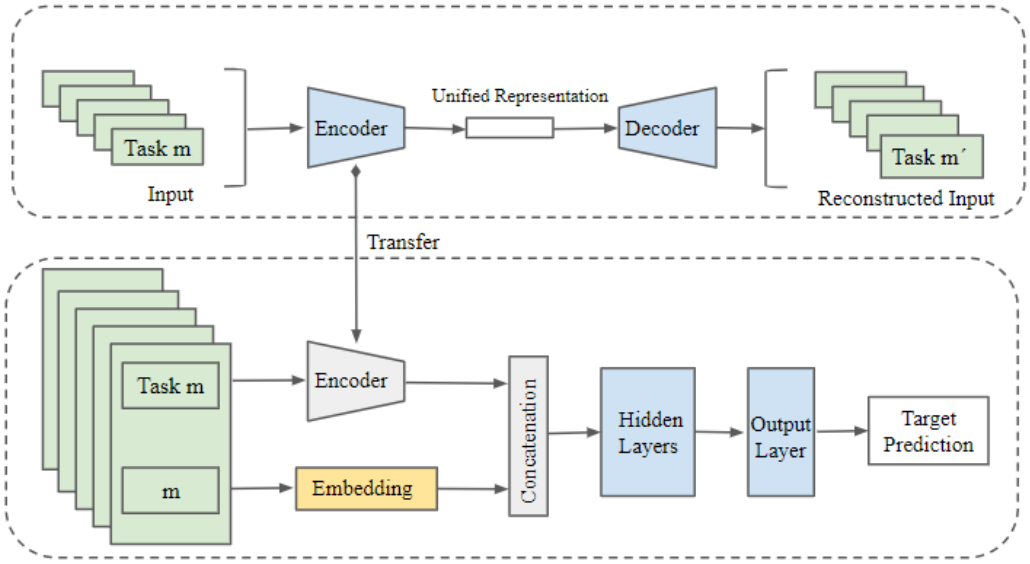


Figure 1. The figure represents the UAE-TENN (Unified Autoencoder-Task-Embedding Neural Network) architecture. In this architecture, the task features are indicated in green, trainable neural network layers in blue, task-specific embedding in orange, and non-trainable layers in grey. The upper portion demonstrates Unified Autoencoder training using multi-task input data. The encoder thus learned is utilized in the lower portion, which represents training a neural network with encoded input and task embedding.

The architecture of the AE network can be conceptualized as the combination of an encoding function f_θ responsible for projecting inputs into a latent space, and a decoding function h_θ focused on reconstructing the original input using the latent space representation. The overarching objective of the AE involves minimizing the discrepancy between the original input and its reconstructed counterpart, as represented in Equation (1). Here, $\mathbf{X}$ signifies the unified data derived from all tasks X^m for $m \in \{1, \ldots, M\}$—and d denotes the mean squared error between the input and its reconstruction. The parameter set θ encompasses the trainable parameters of an AE [8].

$$\min_{\theta} \sum d(\mathbf{X}, h_\theta(f_\theta(\mathbf{X}))) \tag{1}$$

3.2.2. Task-Embedding Neural Network (TENN)

An embedding serves as a technique that transforms discrete categorical values into continuous value vectors. This practice finds its significance in various domains, with

word2vec being a well-recognized instance employed in NLP [19]. Additionally, the broader utility of embeddings in representing general categories through vectorization is also documented [20]. In the context of neural networks, an embedding layer is designed to generate concise, continuous vector representations for discrete variables. This approach offers advantages over using high-dimensional one-hot encodings for the same categories. A notable feature of embeddings lies in their applicability to inductive TL scenarios. They facilitate the incorporation of new tasks into the network by introducing a dedicated embedding vector tailored to the specific task. This contrasts with conventional methods, making the process of expanding the network with new tasks notably smoother. In our study, we leverage an embedding layer to convert diverse discrete task indices into task-specific vector representations. This embedding layer effectively translates each task index m into its corresponding vector representation. We refer to such a neural network architecture featuring this embedding layer, coupled with task-specific vector representations as the Task-Embedding Neural Network (TENN).

Mathematical representation of task embeddings: The mathematical formulation of this entity embedding layer is outlined in [20]. This formulation has been adapted to task embeddings $g(m)$, as demonstrated in [4], and is presented below:

$$g(m) = \sum_{\alpha=1}^{M} \mathbf{w}_{\alpha\beta}\delta_{m\alpha} = \mathbf{w}_{m\beta} \tag{2}$$

Here, $\mathbf{w}_{\alpha\beta}$ signifies the weight matrix that can be learned, and $\delta_{m\alpha}$ represents the Kronecker delta function. This function holds a value of 1 when the values are equal and 0 otherwise.

$$\delta_{m\alpha} = \begin{cases} 1, & \text{if } m = \alpha, \\ 0, & \text{if } m \neq \alpha. \end{cases} \tag{3}$$

In this context, $\delta m\alpha$ is a vector with a length of M, representing M tasks and only the entry corresponding to $\alpha = m$ is non-zero, resembling a one-hot encoding vector. Meanwhile, $\mathbf{w}_{\alpha\beta}$ stands as a weight matrix encompassing all task embeddings. Consequently, the function $g(m)$ effectively translates the discrete task index value into a continuous, real-valued vector.

Practical interpretation of task embeddings: The TE layer can be analogously interpreted as a look-up table, featuring vector values corresponding to each task index. These vectors' elements constitute the learned weights during the course of neural network training. To illustrate, envision a two-dimensional TE vector for each of the M tasks, therefore establishing a TE look-up table with dimensions of $M \times 2$. With each training instance, the specific task index guides the selection of a single row from the look-up table. This chosen row is then concatenated with the instance's other input features. The TE for a given task m can be obtained by taking the dot product of the one-hot encoding vector for tasks with the embeddings look-up table. Specifically, for task m, the m-th row of the look-up table, represented as $[w_{m1}, w_{m2}]$ serves as the corresponding embedding vector.

$$g(m) = [0, 0, \cdots, 1, \cdots, 0] \begin{bmatrix} w_{11}, w_{12} \\ w_{21}, w_{22} \\ \vdots \\ w_{m1}, w_{m2} \\ \vdots \\ w_{M1}, w_{M2} \end{bmatrix} = [w_{m1}, w_{m2}]$$

3.2.3. Training UAE-TENN

We adopt a simple configuration for the UAE architecture, featuring a single hidden encoding layer to capture the encoded representation of input features from all tasks. Subse-

quently, the encoded inputs derived from the UAE are combined with task embeddings and introduced as inputs to a supervised neural network, as illustrated in Figure 1. The training process of UAE involves the utilization of NWP input features from all tasks, represented as $\mathbf{X}$. Consequently, the encoder f_θ learns a shared representation encompassing weather attributes across all parks. This results in the transformation of the k input features within $\mathbf{X}$ into a lower-dimensional latent representation denoted as $f_\theta(\mathbf{X})$. The same architecture with the ReLU activation function and the Adam optimizer is considered for all datasets to maintain uniformity. The other hyperparameters, such as the number of epochs and batch size, are fine-tuned. The dimensionality of the encoding, a key hyperparameter, is carefully chosen to reduce the dimension of input k features to approximately $k/2$ dimensions for each dataset.

In the subsequent stage, a TENN network undergoes training using encoded inputs $f_\theta(\mathbf{X})$, along with task-specific information $g(m)$ from the TE layer. The MTL function encompasses all M tasks and learns from training instances $(\mathbf{x}_i^m, y_i^m, g(m))$, where $m \in \{1, \ldots, M\}$ pertains to tasks, $\mathbf{x}_i^m \in \mathbf{X}$, and $y_i^m \in \mathbf{Y}$. The TENN network is concurrently trained on multiple tasks, with instances from various tasks being present within each batch. This approach facilitates the acquisition of embedding vectors in a manner that fosters similarity between related tasks. The learned TE vectors can also be employed for gauging the similarity among tasks. The task-embedding dimensionality is also tuned and the results are shared in subsequent sections. To ensure a valid comparison across datasets, a consistent neural network structure featuring six hidden layers, Leaky ReLU activation, and the Adam optimizer is employed. Further hyperparameters, including batch size and epochs, are fine-tuned, with additional details provided in the Supplementary Materials.

3.2.4. Inference UAE-TENN

After training UAE-TENN, we derive inferences for each task separately, such that the results can be compared with STL approaches. During inference, as shown in Figure 2, the encoded task inputs $f_\theta(x_i^m)$ and respective TE vectors $g(m)$ are concatenated and passed as input to TENN, to get the task-specific power prediction $\hat{y}_i^m$. The task-specific predictions are then evaluated using the measures described in Section 4.2.

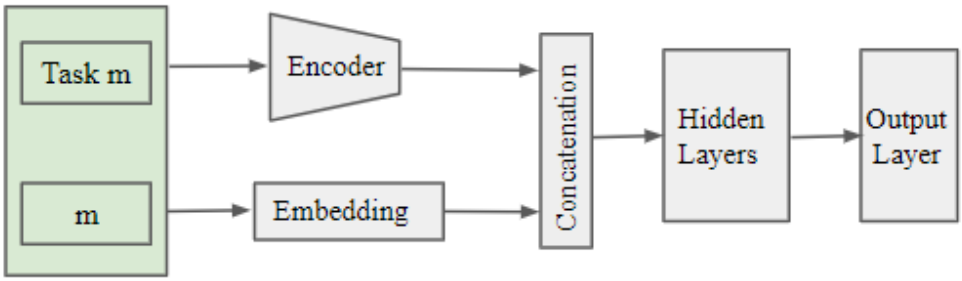


Figure 2. The task-specific inference procedure using UAE-TENN architecture. This picture showcases the inference procedure for task m.

4. Experimental Evaluation

In this section, we outline the datasets used, present the experimental setup, and evaluate the results to address our research questions. The datasets are outlined in Section 4.1 and the evaluation metrics adopted for these experiments are described in Section 4.2. The experimental design and analysis of the distinct experiments for the first three research questions are detailed in Sections 4.3–4.5, respectively.

4.1. Datasets

The experiments are conducted on six renewable-energy power-forecasting datasets, encompassing three pairs of PV and wind datasets derived from various sources, including real-world open-source, synthetic, and private real-world data. We represent real-world

open-source datasets as PO (PV Open) and WO (Wind Open). Similarly, synthetic datasets are represented as PS (PV Synthetic) and WS (Wind Synthetic), and private real-world datasets are named PR (PV Real) and WR (Wind Real), respectively. These datasets are enriched with essential information, typically found in PV or wind parks, consisting of NWP data, historical power-generation data, and metadata [21]. The primary input features for all datasets are NWP-related, while the target variable is day-ahead power generation. In the case of synthetic and real-world data sources, we have additional metadata available for each park comprising physical characteristics such as geographical locations, tilt and azimuth angles of PV panels, turbine type, hub height, rotor diameter, etc for wind parks. The NWP data for PV parks includes features such as solar direct radiation, solar diffuse radiation, temperature, humidity, and solar position, among others. In contrast, wind park NWP data incorporates features like wind speed, wind direction, air pressure, temperature, humidity, and others.

The PV Open (PO) dataset comprises data from 21 parks, while the Wind Open (WO) dataset encompasses data from 45 parks. The PV Synthetic (PS) dataset comprises 118 parks, with an equivalent number of parks randomly selected from the Wind Synthetic (WS) dataset, which consists of data from 263 parks. Further details, including the indices of considered parks in the WS dataset and scatter plots of all datasets, can be found in the Supplementary Materials. The PV Real (PR) dataset encompasses data from 42 parks, while the Wind Real (WR) dataset contains data from 185 parks. Table 1 provides a comprehensive overview of these datasets, offering insights into the number of features, the total parks included in each dataset, and an overview of several samples in source parks and target zero-shot parks. The real-world open-source datasets, denoted as PO (German Solar Farm) and WO (European Wind Farm), are publicly accessible and have been widely used by researchers in the field [4,22]. The synthetic datasets PS and WS are also publicly accessible and are derived from [23]. Notably, the key distinction between these open-source and synthetic datasets is in the source of their Numerical Weather Prediction (NWP) data. The open-source datasets rely on the ECMWF weather model [24], while the synthetic datasets are based on the ICON-EU weather model [25].

Table 1. Datasets overview with source parks and target zero-shot parks information.

Dataset	# Features	# Total Parks	Source Data			Target Zero-Shot Data		
			# Parks	# Mean Train Samples	# Mean Test Samples	# Parks	# Mean Train Samples	# Mean Test Samples
Pv Open (PO)	51	21	17	4553	1518	4	4371	1457
Wind Open (WO)	7	45	36	11,462	3821	9	9287	3096
Pv Synthetic (PS)	14	118	94	8405	4234	24	8158	4225
Wind Synthetic (WS)	24	118	94	8467	4223	24	8466	4288
Pv Real (PR)	19	42	34	59,166	19,722	8	53,892	17,964
Wind Real (WR)	27	185	148	46,405	15,468	37	36,289	12,096

Across all datasets, we have divided the data into source parks and target zero-shot parks using an 80/20 percentage split. The source parks are exclusively used for STL and MTL experiments, while the target zero-shot parks are reserved for the ZSL experiment evaluation. In the synthetic dataset pair (PS and WS), a specific test flag is employed to identify test data. For both the open dataset pairs (PO, WO) and the real dataset pairs (PR, WR), it is important to note that not all source parks have the same amount of historical data available. To address this variation, a split of 75/25 of historical data for training and testing has been applied to each park. This training data covers all seasons to ensure that the UAE's representation learning captures the changing distributions of weather features effectively.

4.2. Evaluation Measures

In all our experiments, we compare the performance metrics of the proposed UAE-TENN with the baseline models specific to each experiment. Our primary evaluation metric is the Root-Mean-Squared Error (RMSE) calculated for each individual park, denoted by RMSE_m as expressed in Equation (4), where y_i^m and $\hat{y}_i^m$ represent the actual and predicted

power-generation values for park m with a total of N data samples. Furthermore, we compute the average RMSE across all M parks in a dataset, which we refer to as aRMSE, as defined in Equation (5).

$$\text{RMSE}_m = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i^m - \hat{y}_i^m)^2} \tag{4}$$

$$\text{aRMSE} = \frac{1}{M}\sum_{m=1}^{M}\text{RMSE}_m \tag{5}$$

We compute a Skill_m metric for each park to assess the performance enhancement achieved by the proposed UAE-TENN compared to the baseline model. The mean skill across all parks within a dataset is denoted as Skill, as indicated in Equation (7). In the results tables, the skill values can be understood as the percentage improvement in performance relative to the baseline model. For instance, a Skill value of 0.245 for the PO dataset in Table 2 translates to an average performance improvement of 24.50% by UAE-TENN in comparison to the baseline model.

$$\text{Skill}_m = 1 - \left(\frac{\text{RMSE}_{\text{reference}_m}}{\text{RMSE}_{\text{baseline}_m}}\right) \tag{6}$$

$$\text{Skill} = \frac{1}{M}\sum_{m=1}^{M}\text{Skill}_m \tag{7}$$

In addition, we provide the standard deviation (Std) of RMSE values across all datasets in the results tables. Lower values for both aRMSE and Std metrics indicate improved performance, while for the skill metric, higher values denote enhanced performance. To assess the statistical significance between a reference model and the baseline, we employ the Wilcoxon one-sided signed-rank test (with a significance level of $\alpha = 0.05$) [26]. If the reference model significantly outperforms the baseline in a dataset, this is denoted by an asterisk symbol (*).

Table 2. Evaluation results of STL methods with UAE-TENN for three different PV power datasets. The asterisk (*) symbol indicates significantly different aRMSE values of reference than baseline (AE-NN) tested through a one-sided Wilcoxon signed-rank test with $\alpha = 0.05$. The bold values in a column indicates best performance. Additionally, it's important to note that both open-source datasets, PO and WO, contain NA values for the PHY model, as there are no available PHY model results.

Model Type	PV aRMSE			PV Skill			PV Std		
	PO	**PS**	**PR**	**PO**	**PS**	**PR**	**PO**	**PS**	**PR**
AE-NN	0.112	0.094 *	0.124	0.000	0.000	0.000	0.022	0.015	0.019
PHY	NA	0.102	0.189	NA	−0.084	−0.587	NA	0.016	0.199
LR	0.088 *	0.104	0.116 *	0.218	−0.096	0.064	0.022	0.019	0.019
GBRT	0.085 *	0.092	0.109 *	0.239	0.023	0.120	0.021	0.014	0.019
NN	0.087 *	0.092 *	0.112 *	0.221	0.021	0.094	0.023	0.013	0.020
UAE-TENN	**0.084 ***	**0.083 ***	**0.095 ***	**0.245**	**0.118**	**0.222**	**0.020**	**0.013**	**0.011**
Mean UAE-TENN		0.087			0.195			0.014	

4.3. Comparison with STL Methods

In this section, we conduct an experiment to answer the following research question:

Question 1: Does unified encoding of task-independent features along with task embeddings decrease the forecast error for PV and wind parks compared to STL methods?

This experiment involves the evaluation of five distinct single-task models for comparison: Physical (PHY), Linear Regression (LR), Gradient Boosting Regression Trees (GBRT), Neural Network (NN), and Autoencoder-Neural Network (AE-NN). The PHY model generates power forecasts for PV or wind parks based on their physical characteristics and respective power curves, and it does not rely on historical data [23]. Notably, while the synthetic and real datasets incorporate predictions from the physical model, the open-source datasets do not. These single-task models serve as benchmarks, offering a range of modeling techniques spanning statistical approaches, tree-based methods, and neural networks. The NN model represents a basic feedforward multi-layer neural network architecture, which is included for comparison purposes with other single-task models like LR and GBRT. The AE-NN model, on the other hand, involves training an autoencoder (AE) on each park individually and subsequently training a neural network (NN) on the encoded lower-dimensional inputs. This AE-NN model is included to assess whether single-task representation learning enhances performance relative to other single-task models, and it serves as the baseline for comparison. This AE-NN model is chosen as a baseline because AE-NN is a single-task alternative to the proposed multi-task UAE-TENN model. Therefore, comparing against the AE-NN model provides a fair assessment of the effectiveness of our approach. This is also similar to the baseline models considered in recent studies [14,27] All single-task models are compared against the proposed UAE-TENN method, and their evaluations are conducted separately for each task. As a preliminary insight into the interrelationships among parks within a dataset, we compute the standard deviation of the unified target data across all parks within each dataset. The calculated standard deviation values for the datasets PO, WO, PS, WS, PR, and WR are, respectively, 0.225, 0.242, 0.198, 0.256, 0.227, and 0.244. This assessment suggests that, intrinsically, wind datasets exhibit greater variability in power generation across parks compared to PV datasets.

Tables 2 and 3 present the evaluation results of STL models in comparison with the proposed UAE-TENN method across all datasets. Notably, the proposed method consistently achieves the lowest average Root-Mean-Squared Error (aRMSE) and significantly outperforms the baseline models in all PV datasets. However, among the wind datasets, the lowest aRMSE is attained only in the WR dataset. This distinction arises due to the inherent dissimilarity among wind parks in contrast to the more homogeneous characteristics of solar parks, as previously mentioned. The Skill metric reveals that, on average, the proposed UAE-TENN method enhances performance by 19.5% for PV parks and 5.7% for wind parks. Specifically, the Skill score is positive for all PV datasets and two out of three wind datasets. The negative Skill score observed for all methods in the WS dataset suggests that the representation learning achieved by the AE-NN baseline in the STL setting surpasses the performance of other methods. It is worth noting that, on average, the standard deviation (std) values are lower for PV datasets compared to wind datasets. However, it is essential to highlight that the proposed method exhibits comparatively weaker performance in the WS dataset compared to the baseline model. This discrepancy can be attributed to the limited commonalities among the wind parks in this dataset, making it challenging to establish a shared representation using the Unified Autoencoder (UAE) as employed in the proposed method. Furthermore, the higher aRMSE values and negative Skill scores observed for the AE-NN, LR, and NN methods in the WR dataset are likely due to the presence of outliers, as evident from the corresponding higher std values for these methods in WR. Figure 3 presents a comparison of aRMSE values for STL methods across the three PV datasets. The aRMSE value for the PHY model in the PO dataset is not applicable (NA), as mentioned earlier open datasets do not have PHY model predictions. However, for illustrative purposes in this figure, it has been substituted with the corresponding value from the PS dataset.

Table 3. Evaluation results of STL methods with UAE-TENN for three different wind power datasets. The asterisk (*) symbol indicates significantly different aRMSE values of reference than baseline (AE-NN) tested through a one-sided Wilcoxon signed-rank test with $\alpha = 0.05$. The bold values in a column indicate the best performance.

Model Type	Wind aRMSE			Wind Skill			Wind Std		
	WO	WS	WR	WO	WS	WR	WO	WS	WR
AE-NN	0.132	0.141	6.124	0.000	**0.000**	0.000	0.040	0.054	65.143
PHY	NA	0.255	0.201	NA	−1.259	−0.181	NA	0.095	0.069
LR	0.123	0.156	1.515 *	0.042	−0.355	−0.007	0.031	0.047	10.234
GBRT	0.112 *	0.144	0.158 *	0.133	−0.235	0.088	0.032	0.041	0.043
NN	**0.110** *	**0.137** *	2.598 *	**0.145**	−0.041	−0.037	**0.031**	**0.039**	18.377
UAE-TENN	0.117 *	0.142	**0.142** *	0.096	−0.078	**0.153**	0.032	0.041	**0.029**
Mean UAE-TENN		0.133			0.057			0.034	

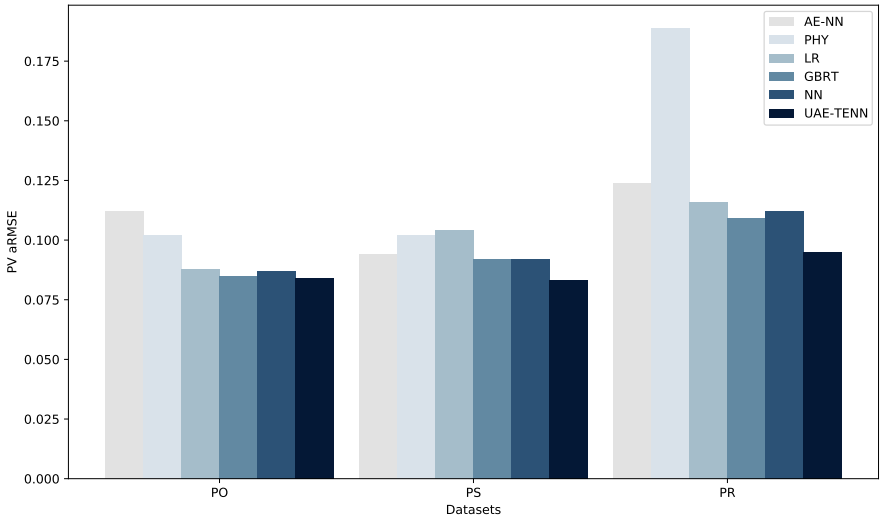


Figure 3. Comparison of aRMSE values among STL methods across three PV datasets PO, PS and PR.

4.4. Comparison with MTL Methods

In this section, we conduct an experiment to answer the following research question:

Question 2: Can autoencoder-based MTL architecture with TE perform better than using either UAE or TE in MTL architecture?

The first MTL approach, UAE-NN, involves feeding encoded inputs from multiple tasks into a neural network, training a single model for all source tasks. The second approach is TENN, where task embeddings are appended to the original inputs of respective tasks, and a neural network is trained accordingly. The training and inference procedures for UAE-NN and TENN architectures mirror the UAE-TENN process. UAE-NN does not include the TE component, while TENN omits the UAE component. Thus, this experiment serves as an ablation study, evaluating the impact of the UAE and TE components in the UAE-TENN architecture. Here, UAE-NN is considered to be the baseline model for comparison.

Tables 4 and 5 demonstrate that UAE-TENN consistently outperforms the baseline UAE-NN, as indicated by positive skill values. UAE-TENN achieves the highest skill scores in five out of six datasets. On average, UAE-TENN exhibits improvements of approximately 10.80% and 22.1% for PV and wind datasets, respectively, compared to the baseline UAE-NN. The ablation study reveals that the TE component contributes more significantly to performance enhancement than the UAE component. This is evident from consistently lower aRMSE values in TENN in both PV and wind datasets. It can be inferred that the collaborative use of both components yields superior results compared to employing either component in isolation. An illustrative time series plot showcasing power forecasts for a

PV park from the PR dataset is presented in Figure 4 and the comparison of aRMSE for PV parks is shown in Figure 5a.

Table 4. Evaluation results of MTL methods with UAE-TENN for three PV datasets with baseline UAE-NN as a baseline model. The asterisk (*) symbol indicates significantly different aRMSE values of reference than baseline (AE-NN) tested through a one-sided Wilcoxon signed-rank test with $\alpha = 0.05$. The bold values in a column indicate the best performance.

Model Type	PV aRMSE			PV Skill			PV Std		
	PO	PS	PR	PO	PS	PR	PO	PS	PR
UAE-NN	0.088	0.107	0.105	0.000	0.000	0.000	0.024	0.021	0.020
TENN	0.087	0.091 *	0.096 *	−0.006	0.136	0.080	0.020	0.015	0.014
UAE-TENN	**0.084** *	**0.083** *	**0.095** *	**0.029**	**0.209**	**0.087**	**0.020**	**0.013**	**0.011**
Mean UAE-TENN		0.087			0.108			0.014	

Table 5. Evaluation results of MTL methods with UAE-TENN for three wind datasets with baseline UAE-NN as a baseline model. The asterisk (*) symbol indicates significantly different aRMSE values of reference than baseline (AE-NN) tested through a one-sided Wilcoxon signed-rank test with $\alpha = 0.05$. The bold values in a column indicate the best performance.

Model Type	Wind aRMSE			Wind Skill			Wind Std		
	WO	WS	WR	WO	WS	WR	WO	WS	WR
UAE-NN	0.156	0.199	0.175	0.000	0.000	0.000	0.036	0.083	0.045
TENN	**0.115** *	0.146 *	0.148 *	**0.267**	0.218	0.140	**0.031**	**0.040**	0.030
UAE-TENN	0.117 *	**0.142** *	**0.142** *	0.253	**0.240**	**0.173**	0.032	0.041	**0.029**
Mean UAE-TENN		0.133			0.222			0.034	

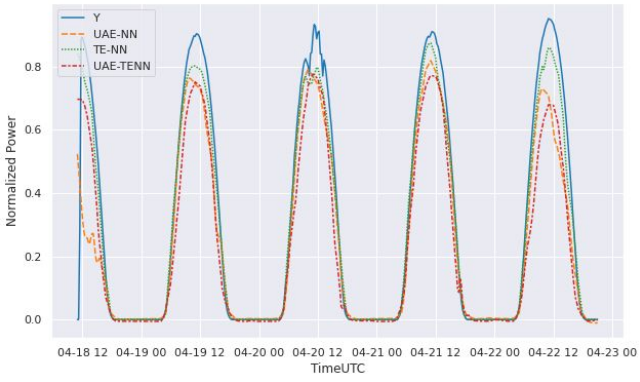


Figure 4. Exemplary power forecast comparison of MTL methods for a PV park from the real-world dataset. The X-axis represents the time and Y axis represents the normalized power generation. The solid blue line (Y) represents the actual power generated by the park and the other three dotted lines represent the predicted values by different MTL models.

4.5. Zero-Shot Learning Experiment

In this section, we conduct an experiment to answer the following research question:

Question 3: Are autoencoder-based MTL architectures with TE capable of providing better forecasts in the ZSL scenario for renewable-power forecasts compared to the baseline model?

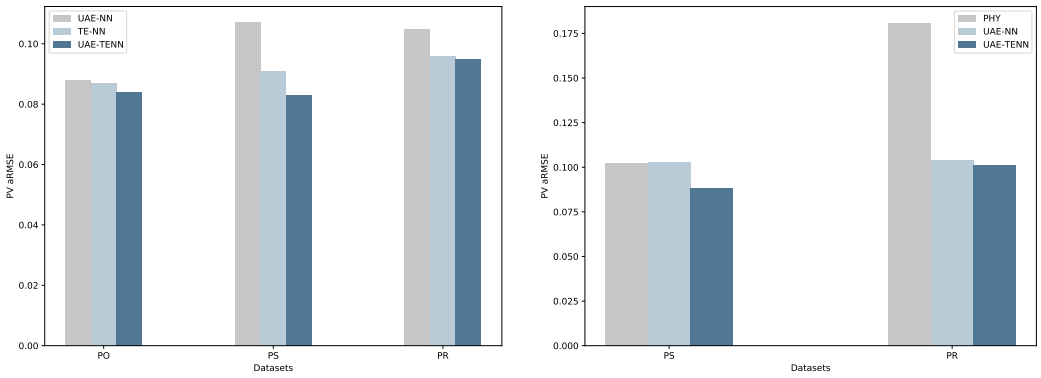
In the context of zero-shot learning (ZSL), where historical data for the target task is absent, predictions are reliant on park metadata. This metadata encompasses various details such as the geographical location, tilt angle, azimuth angle, elevation, etc., for PV parks, and for wind parks, it encompasses parameters like hub height, turbine diameter, number of generators, elevation, and geographical location.

In this experiment, we compare the performance of the proposed method with the PHY model and the UAE-NN model. These models were selected from the experiments conducted in Sections 4.3 and 4.4 due to their applicability to the zero-shot learning (ZSL) experimental setup. The objective is to predict the power generation of a new park without access to target task data. The PHY model serves as the baseline for comparison. We evaluate the proposed UAE-TENN architecture outlined in previous sections for this experiment. However, as mentioned in Section 3.2.4, predicting using the UAE-TENN architecture requires a task-embedding vector for the specific task. As the new target park lacks a task-embedding vector, we address this by substituting it with the task-embedding vector of the most similar park. To identify the most similar source park, we explore three similarity metrics based on park metadata: geographical distance, cosine distance between metadata features, and Euclidean distance between t-SNE embeddings of metadata features [28]. It is important to note that the open-source datasets (PO and WO) lack metadata information for parks, preventing us from selecting a suitable source task based on metadata. Consequently, this experiment is conducted exclusively on the 20% of parks designated as target zero-shot parks in the PV datasets (PS and PR) and the wind datasets (WS and WR), where such metadata are available.

The results presented in Table 6 indicate that the UAE-NN model outperforms the PHY baseline model in the PR and WR datasets, as demonstrated by positive skill values. The proposed UAE-TENN method demonstrates an average performance improvement of 19.60% for PV parks. However, for wind parks, the negative skill value arises from the limited similarity among parks within a dataset, making the TE imputation from the most similar park less beneficial. This leads to the observation that the TE component proves advantageous when a strong relationship exists between tasks. It is noteworthy to consider the intrinsic diurnal cycle of solar energy, which facilitates ZSL for new parks even when historical solar radiation data are absent, thanks to the geographical location. Conversely, ZSL for wind parks is more challenging due to higher wind speed variance across locations, making power prediction difficult without historical wind speed data. Another interesting observation pertains to the similarity metrics. Specifically, on the real dataset pair (PR and WR), leveraging only geographical distance yields superior results. However, for the synthetic dataset pair, incorporating all metadata features, including geographical information, yields better performance outcomes. The aRMSE comparison of this experiment on PV datasets is illustrated in Figure 5b.

Table 6. Evaluation results of ZSL methods with UAE-TENN for four datasets with PHY as a baseline model. The asterisk (*) symbol indicates significantly different aRMSE values of reference than baseline (AE-NN) tested through a one-sided Wilcoxon signed-rank test with $\alpha = 0.05$. The bold values in a column indicate the best performance.

Model Type	PV aRMSE		Wind aRMSE		PV Skill		Wind Skill		PV Std		Wind Std	
	PS	PR	WS	WR	PS	PR	WS	WR	PS	PR	WS	WR
PHY	0.102	0.181	**0.212**	0.187	0.000	0.000	**0.000**	0.000	**0.018**	0.156	**0.047**	0.061
UAE-NN	0.103	0.104	0.213 *	**0.172** *	−0.009	0.226	−0.076	**0.049**	0.026	0.017	0.138	**0.039**
UAE-TENN	**0.088** *	**0.101** *	0.220	0.180	**0.145**	**0.247**	−0.110	−0.020	0.024	**0.016**	0.159	0.042
Mean UAE-TENN	0.094		0.200		0.196		−0.065		0.020		0.100	



(**a**) (**b**)

Figure 5. MTL and ZSL comparison plots. (**a**) Comparison of aRMSE values among MTL methods across three PV datasets PO, PS, and PR. (**b**) Comparison of aRMSE values among ZSL methods across three PV datasets PO, PS, and PR.

5. Evaluating the Impact of Convolutional and LSTM Autoencoders on Architecture Performance

In this section, we present extension architectures for the architecture proposed in our previous work [29], which is introduced in Section 3.2. Specifically, we incorporate convolutional and LSTM autoencoders to encode input data, replacing the regular autoencoder. We provide a detailed overview of the architecture and experimental evaluation in this section.

5.1. Unified Convolutional Autoencoder-Task Embedding (UCAE-TENN)

In this section, we conduct an experiment to evaluate and compare the convolutional autoencoder with the regular fully connected autoencoder in the proposed architecture. In this experiment, we replaced the fully connected neural network layers with convolutional layers in the encoder part of the architecture of Figure 1. In the convolutional autoencoder architecture, a stack of three convolutional 1D layers is utilized for encoding the data, and the same three convolutional transpose layers are used for the decoder part of the architecture. The convolutionally encoded data are then appended with the regular two-dimensional task-embedding layer to train a neural network. The task embeddings are learned along with other neural network parameters like in the previous UAE-TENN architecture.

For the zero-shot learning experiment, the task embedding of the nearest neighboring park based on the t-SNE embeddings of metadata is utilized for predicting the zero-shot target park as described in Section 4.5.

5.2. Unified LSTM Autoencoder-Task Embedding (ULAE-TENN)

In this section, we conduct an experiment to evaluate and compare the LSTM (Long Short-Term Memory) autoencoder with the regular fully connected autoencoder and the convolutional autoencoder. For the LSTM autoencoder architecture, we use two LSTM layers in the encoder part of the architecture and use the same two LSTM layers for decoding. To enable training using an LSTM autoencoder, we convert the data to temporal data by adding an extra dimension based on the lookback of the time series. The three-dimensional temporal data are then used for training an LSTM autoencoder, and the encoded data are appended with the task-embedding layer, as in the previous architectures. The task embeddings are learned while training the neural network and utilized for inference. For the zero-shot learning scenario, we utilize the task embedding of the nearest park for the new zero-shot park, and a prediction is made following the same inference methodology as previously mentioned.

5.3. Experimental Evaluation

As an extension of the previous experiments, we consider and evaluate the performances of utilizing convolutional and LSTM layers in the autoencoders of the proposed architecture. The experiments are performed only on the photovoltaic datasets as experiments in previous sections showed good performance improvement for photovoltaic datasets compared to wind datasets. In this section, we conduct an experiment to answer the following research question:

Question 4: Can convolutional and LSTM autoencoders improve the performance compared to regular fully connected layers for STL power forecast of photovoltaic datasets?

The experimental results of the comparison between convolutional and LSTM autoencoders with regular fully connected layers for STL inference are presented in Table 7. The findings suggest that UAE-TENN outperforms the other two complex models that use convolutional and LSTM layers in UCAE-TENN and ULAE-TENN, respectively. Although the convolutional autoencoder model shows the same aRMSE as UAE-TENN for the PR dataset, its skill metric is lower. Additionally, the convolutional autoencoder model outperforms the other two models in terms of standard deviation metric for both PO and PR datasets. Conversely, for all three metrics across all three datasets, the ULAE-TENN model with LSTM autoencoder exhibits poor performance, indicating that the temporal encoding does not contribute to improving the power forecast for these photovoltaic datasets. The superior performance of the regular fully connected layers in the UAE-TENN model, as demonstrated by a high skill metric for all three datasets, suggests that simple encoding of information is adequate for these datasets.

Table 7. Comparison of regular, convolutional, and LSTM autoencoders in the proposed architecture for three different datasets in STL inference evaluation. The asterisk (*) symbol indicates significantly different aRMSE values of reference than baseline (AE-NN) tested through a one-sided Wilcoxon signed-rank test with $\alpha = 0.05$. The bold values in a column indicate the best performance.

Model Type	PV aRMSE			PV Skill			PV Std		
	PO	PS	PR	PO	PS	PR	PO	PS	PR
UAE-TENN	**0.085 ***	**0.084 ***	**0.111 ***	**0.236**	**0.108**	**0.100**	0.021	**0.013**	0.018
UCAE-TENN	0.088 *	0.091 *	**0.111 ***	0.213	0.032	0.097	**0.019**	0.014	**0.111**
ULAE-TENN	0.108	0.114	0.138	0.029	−0.216	−0.117	0.023	0.022	0.138

The superior performance of the UAE-TENN model, which uses regular fully connected layers, compared to the UCAE-TENN and ULAE-TENN models can be attributed to the nature of the PV power-generation dataset. Unlike image and speech datasets, which often require convolutional and LSTM layers for effective feature extraction, the PV power-generation data are relatively simple and can be effectively represented by simple encoding of information. Furthermore, the UAE-TENN model has fewer parameters to learn, making it less prone to overfitting compared to the UCAE-TENN and ULAE-TENN models with more complex architectures. Additionally, the simple encoding of information in the UAE-TENN model can allow for more efficient training, which can result in better generalization performance.

Zero-Shot Learning Scenario

The experimental results for the following research question on ZSL is also evaluated and the results are displayed in Table 8.

Question 5: Do convolutional and LSTM autoencoders do a better power forecast in zero-shot learning scenarios compared to unified regular autoencoders in the proposed architecture?

Table 8. Comparison of results for ZSL methods on two datasets with PHY as a baseline model. The asterisk (*) symbol indicates significantly different aRMSE values of reference than baseline (AE-NN) tested through a one-sided Wilcoxon signed-rank test with $\alpha = 0.05$. The bold values in a column indicate the best performance.

Model Type	PV aRMSE		PV skill		PV Std	
	PS	PR	PS	PR	PS	PR
UAE-TENN	**0.085**	0.108 *	**0.159**	0.194	**0.015**	0.037
UCAE-TENN	0.091	**0.106** *	0.101	**0.210**	0.019	0.042
ULAE-TENN	0.110	0.135	−0.087	0.018	0.022	**0.035**

The experimental results of zero-shot learning comparison among three different autoencoder models with regular fully connected, convolutional, and LSTM layers are presented in Table 8. The results indicate that the ULAE-TENN model with LSTM layers in autoencoder is performing poorly even in the zero-shot learning scenario, which is consistent with its poor performance in the STL inference evaluation. UAE-TENN and UCAE-TENN models are both performing better than the other in two different datasets. Specifically, for the synthetic dataset PS, the regular autoencoder model UAE-TENN outperforms the other two models in all three metrics. For the real dataset PR, the convolutional autoencoder UCAE-TENN performs better with higher skill and lower aRMSE. Moreover, the LSTM autoencoder model shows better performance than the other two models only in the standard deviation metric for the PR dataset. These results suggest that for these particular datasets, the simpler regular autoencoder model with fully connected layers is generally sufficient even for zero-shot learning park power forecast and performs better than the more complex models with convolutional and LSTM layers. It is also interesting to note aRMSE of none of the three models for the PS dataset is significantly better than the baseline model whereas, for the PR dataset, two of them are significantly better than the baseline model.

6. Influence of Task-Embedding Dimensionality on Performance

In this section, we describe the experiment performed to evaluate the following research question:

Question 6: Does increase in the number of task-embedding dimensions in the proposed architecture improve the performance for different autoencoder models?

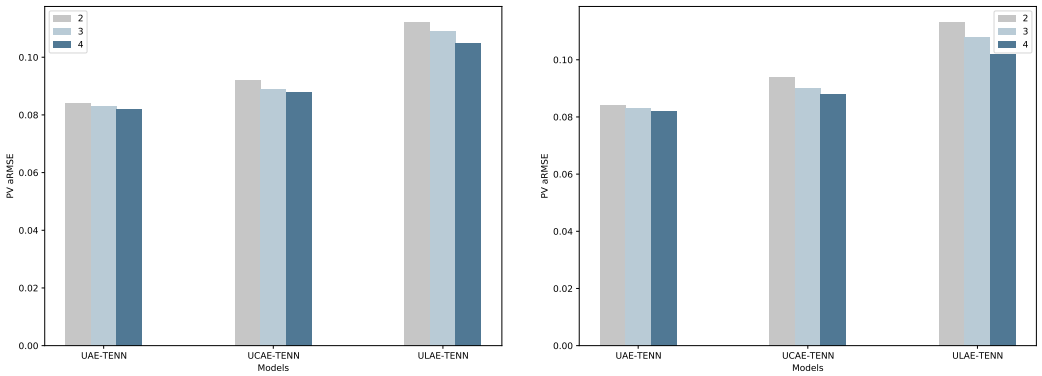
The architecture proposed in Section 3.2 incorporates a task-embedding layer with two dimensions, but it is possible to increase the dimensions of the task-embedding layer. This experiment was conducted to investigate whether an increase in task-embedding dimensions would contribute to performance improvement. The results of this experiment are presented in Tables 9 and 10, which separately evaluate the performance of the proposed architecture in STL inference and ZSL scenarios, respectively. The results from both tables demonstrate that increasing the task-embedding dimension from 2 to 4 consistently lowers the aRMSE values for all three unified architecture models. The same trend is observed in the aRMSE values for the ZSL scenario, as shown in Table 10. However, only two datasets were evaluated in this experiment, as metadata for the PO dataset was unavailable for ZSL experimental evaluation. Overall, the experiment suggests that an increase in task-embedding dimensions can lead to improved performance in the proposed architecture. The comparison of aRMSE across different models for PS dataset in MTL and ZSL scenarios are showcased in Figure 6a,b.

Table 9. Impact of the number of embedding dimensions on aRMSE for three model architectures across two datasets PS and PR in STL inference evaluation.

Model Type	UAE-TENN		UCAE-TENN		ULAE-TENN	
	PS	**PR**	**PS**	**PR**	**PS**	**PR**
2	0.084	0.111	0.092	0.113	0.112	0.137
3	0.083	**0.110**	0.089	**0.111**	0.109	0.137
4	**0.082**	**0.110**	**0.088**	**0.111**	**0.105**	**0.136**

Table 10. Impact of the number of embedding dimensions on aRMSE for three model architectures across two datasets PS and PR in ZSL scenario evaluation.

Model Type	UAE-TENN		UCAE-TENN		ULAE-TENN	
	PS	**PR**	**PS**	**PR**	**PS**	**PR**
2	0.084	0.108	0.094	0.108	0.113	0.127
3	0.083	0.107	0.090	0.107	0.108	**0.126**
4	**0.082**	**0.106**	**0.088**	**0.106**	**0.102**	**0.126**

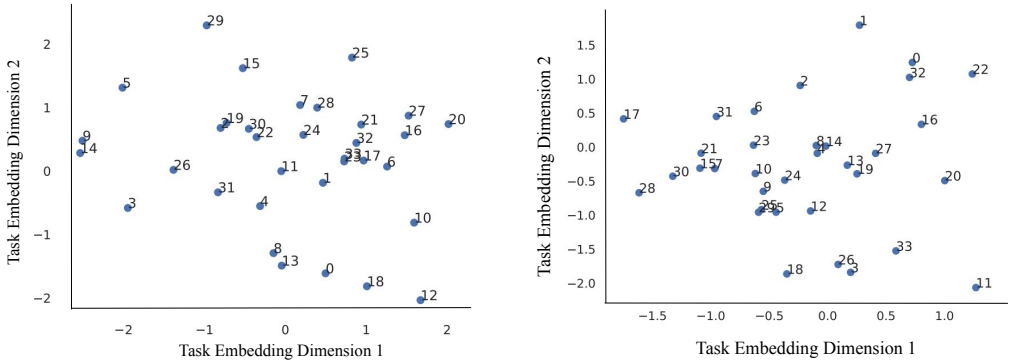


(**a**) (**b**)

Figure 6. MTL and ZSL comparison plots. (**a**) Comparison of aRMSE values for MTL with varying task embedding dimensions across three unified autoencoder models for the PS dataset. (**b**) Comparison of aRMSE values for ZSL with varying task embedding dimensions across three unified autoencoder models for the PS dataset

7. Utility of Task Embeddings for Visualization

The neural network in the proposed architecture is trained to learn task embeddings. The learned task embeddings can be visualized for better understanding and interpretability of the relationship among the tasks. The interpretation of such task embeddings is particularly easy for two-dimensional embeddings based on visualization plots. The plots in Figure 7a,b demonstrate that different models learn task embeddings differently for the same dataset due to random initialization and different encoding mechanisms of fully connected layers, convolutional layers, and LSTM layers across these three architectures, respectively. It can be interpreted that the distance among these learned task embeddings denotes the similarity among tasks. This task-embedding plot also can be utilized for task clustering as it reveals the relationship among tasks to identify patterns and groups. The clusters of closely positioned task embeddings might indicate groups of related tasks that share similar underlying dynamics. Conversely, isolated embeddings can signify tasks with unique properties or distinct behaviors. The embedding distances indicate task similarity or dissimilarity providing indirect information about task relationships. The changes in embeddings over time or conditions offer insights into evolving task dynamics and identify anomalies.



(**a**) (**b**)

Figure 7. Task embeddings visualization plots. (**a**) Visualization of two-dimensional task embeddings learned from UCAE-TENN architecture for PR dataset. (**b**) Visualization of two-dimensional task embeddings learned from ULAE-TENN architecture for PR dataset.

8. Conclusions and Future Work

In this paper, we have presented a novel MTL architecture that utilizes a combination of simple and effective UAE and TE methods to learn task-independent common representations as well as task-dependent embeddings. We have successfully evaluated the proposed UAE-TENN method for PV and wind day-ahead power forecasting across six datasets and 529 parks. Our experimental findings demonstrate that, for PV datasets, the proposed method outperforms both STL and MTL baseline models. However, in the wind datasets, owing to the absence of robust temporal correlations, higher standard deviations, and an increased presence of outliers, the proposed method does not exhibit notably superior performance. This observation holds true even in the context of zero-shot learning (ZSL), where the proposed method notably enhances performance for PV parks when contrasted with wind parks, as the multi-task representation learned by the autoencoder is effective when tasks exhibit strong correlations.

Additionally, we compared the performance of regular autoencoders with convolutional and LSTM variants of autoencoders for input data encoding and found that the regular autoencoder exhibits superior performance for this application. Furthermore, we evaluated the impact of task-embedding dimensionality on model performance and found that increasing the dimensions of task embeddings consistently led to performance improvements, regardless of the autoencoder architecture employed. We also demonstrated the utility of task embeddings for visualization and interpretation. In future research work, we plan to explore the utility of Bayesian task embeddings for MTL architectures. We firmly believe that our proposed MTL framework has the potential to extend its utility to various domains encompassing STL, MTL, and ZSL applications.

Supplementary Materials: The following supporting information can be downloaded at: https://www.mdpi.com/article/10.3390/make5030062/s1. Figure S1: Scatter plots of solar radiation and wind speed features with historical power measurements of a sample park from all PV and wind datasets. Table S1: Hyperparameters tuned for unified autoencoder models—UAE, UCAE, ULAE on six different datasets.

Author Contributions: Conceptualization, C.P.N., S.V. and B.S.; Data curation, C.P.N. and S.V.; Formal analysis, C.P.N.; Funding acquisition, B.S.; Investigation, C.P.N.; Methodology, C.P.N. and S.V.; Project administration, S.V. and B.S.; Resources, B.S.; Software, C.P.N. and S.V.; Supervision, S.V. and B.S.; Validation, C.P.N.; Visualization, C.P.N.; Writing—original draft, C.P.N.; Writing—review and editing, C.P.N., S.V. and B.S. All authors have read and agreed to the published version of the manuscript.

Funding: This work is supported by the Digital-Twin-Solar (03EI6024E) research project funded by the BMWi (German Federal Ministry for Economic Affairs and Energy). Enercast GmbH has provided the real-world datasets.

Data Availability Statement: The open-source datasets PO and WO are accessible at https://www.uni-kassel.de/eecs/en/sections/intelligent-embedded-systems/downloads, accessed on 25 August 2023. The synthetic datasets PS and WS are available at http://doi.org/10.48662/daks-11, accessed on 25 August 2023.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

NWP	Numerical Weather Prediction
PV	PhotoVoltaic
AE	AutoEncoder
UAE	Unified Autoencoder
TE	Task Embedding
TENN	Task-Embedding Neural Network
UAE-TENN	Unified Autoencoder and Task-Embedding Neural Network
LSTM	Long Short-Term Memory Network
TL	Transfer Learning
STL	Single-Task Learning
MTL	Multi-Task Learning
ZSL	Zero-Shot Learning
HPS	Hard Parameter Sharing
SPS	Soft Parameter Sharing
PO	Photovoltaic Open-source dataset
WO	Wind Open-source dataset
PS	Photovoltaic Synthetic dataset
WS	Wind Synthetic dataset
PR	Photovoltaic Real-world dataset
WR	Wind Real-world dataset
RMSE	Root-Mean-Squared Error
aRMSE	Average Root-Mean-Squared Error
Std	Standard deviation
PHY	Physical Model
LR	Linear Regression
GBRT	Gradient Boosting Regression Tree
NN	Neural Network
AE-NN	Autoencoder-Neural Network
UCAE	Unified Convolutional Autoencoder
ULAE	Unified Long short-term memory Autoencoder

References

1. A Global Roadmap for Accelerated sdg7 Action in Support of the 2030 Agenda for Sustainable Development and the Paris Agreement on Climate Change. Available online: https://www.un.org/en/page/global-roadmap (accessed on 18 September 2023).
2. Alkhayat, G.; Mehmood, R. A review and taxonomy of wind and solar energy forecasting methods based on deep learning. *Energy AI* **2021**, *4*, 100060. [CrossRef]
3. Schwartz, R.; Dodge, J.; Smith, N.A.; Etzioni, O. Green AI. *arXiv* **2020**, arXiv:1907.10597. [CrossRef]
4. Schreiber, J.; Vogt, S.; Sick, B. Task embedding temporal convolution networks for transfer learning problems in renewable power time series forecast. In Proceedings of the Joint European Conference on Machine Learning and Knowledge Discovery in Databases, Bilbao, Spain, 13–17 September 2021; Springer: Berlin/Heidelberg, Germany, 2021; pp. 118–134.
5. Zhang, Y.; Yang, Q. A survey on multi-task learning. *IEEE Trans. Knowl. Data Eng.* 2021. *34*, 5586–5609. [CrossRef]
6. Jiao, R.; Huang, X.; Ma, X.; Han, L.; Tian, W. A model combining stacked autoencoder and back propagation algorithm for short-term wind power forecasting. *IEEE Access* **2018**, *6*, 17851–17858. [CrossRef]
7. Zhuang, F.; Qi, Z.; Duan, K.; Xi, D.; Zhu, Y.; Zhu, H.; Xiong, H.; He, Q. A comprehensive survey on transfer learning. *Proc. IEEE* **2020**, *109*, 43–76. [CrossRef]
8. Charte, D.; Charte, F.; del Jesus, M.J.; Herrera, F. An analysis on the use of autoencoders for representation learning: Fundamentals, learning task case studies, explainability and challenges. *Neurocomputing* **2020**, *404*, 93–107. [CrossRef]

9. Ghifary, M.; Kleijn, W.B.; Zhang, M.; Balduzzi, D. Domain generalization for object recognition with multi-task autoencoders. In Proceedings of the IEEE International Conference on Computer Vision, Santiago, Chile, 7–13 December 2015; pp. 2551–2559. [CrossRef]
10. Epstein, B.; Meir, R.; Michaeli, T. Joint auto-encoders: A flexible multitask learning framework. *arXiv* **2017**, arXiv:1705.10494. [CrossRef]
11. Zhuang, F.; Luo, D.; Jin, X.; Xiong, H.; Luo, P.; He, Q. Representation learning via semi-supervised autoencoder for multi-task learning. In Proceedings of the 2015 IEEE International Conference on Data Mining, Atlantic City, NJ, USA, 14–17 December 2015; pp. 1141–1146. [CrossRef]
12. Xu, W.; Li, S.; Lu, Y. Usr-mtl: An unsupervised sentence representation learning framework with multi-task learning. *Appl. Intell.* **2021**, *51*, 3506–3521. [CrossRef]
13. Qureshi, A.S.; Khan, A.; Zameer, A.; Usman, A. Wind power prediction using deep neural network based meta regression and transfer learning. *Appl. Soft Comput.* **2017**, *58*, 742–755. [CrossRef]
14. Liu, X.; Cao, Z.; Zhang, Z. Short-term predictions of multiple wind turbine power outputs based on deep neural networks with transfer learning. *Energy* **2021**, *217*, 119356. [CrossRef]
15. Qureshi, A.S.; Khan, A. Adaptive transfer learning in deep neural networks: Wind power prediction using knowledge transfer from region to region and between different task domains. *Comput. Intell.* **2019**, *35*, 1088–1112. [CrossRef]
16. Wang, L.; Tao, R.; Hu, H.; Zeng, Y.R. Effective wind power prediction using novel deep learning network: Stacked independently recurrent autoencoder. *Renew. Energy* **2021**, *164*, 642–655. [CrossRef]
17. Chen, J.; Zhu, Q.; Li, H.; Zhu, L.; Shi, D.; Li, Y.; Duan, X.; Liu, Y. Learning heterogeneous features jointly: A deep end-to-end framework for multi-step short-term wind power prediction. *IEEE Trans. Sustain. Energy* **2019**, *11*, 1761–1772. [CrossRef]
18. Ruder, S. An overview of multi-task learning in deep neural networks. *arXiv* **2017**, arXiv:1706.05098. [CrossRef]
19. Mikolov, T.; Sutskever, I.; Chen, K.; Corrado, G.S.; Dean, J. Distributed Representations of Words and Phrases and Their Compositionality. *Adv. Neural Inf. Process. Syst.* **2013**, *26*. Available online: https://proceedings.neurips.cc/paper_files/paper/2013/file/9aa42b31882ec039965f3c4923ce901b-Paper.pdf (accessed on 5 August 2023).
20. Guo, C.; Berkhahn, F. Entity embeddings of categorical variables. *arXiv* **2016**, arXiv:1604.06737. [CrossRef]
21. Nivarthi, C.P. Transfer learning as an essential tool for digital twins in renewable energy systems. *arXiv* **2022**, arXiv:2203.05026. [CrossRef]
22. Menezes, D.; Mendes, M.; Almeida, J.A.; Farinha, T. Wind Farm and Resource Datasets: A Comprehensive Survey and Overview. *Energies* **2020**, *13*, 4702. [CrossRef]
23. Vogt, S.; Schreiber, J. Synthetic Photovoltaic and Wind Power Forecasting Data. *arXiv* **2022**, arXiv.2204.00411. [CrossRef]
24. European Centre for Medium-Range Weather Forecasts. 2020. Available online: http://www.ecmwf.int/ (accessed on 25 August 2023).
25. Icosahedral Non-Hydrostatic-European Union. 2023. Available online: https://www.dwd.de/DWD/forschung/nwv/fepub/icon_database_main.pdf (accessed on 25 August 2023).
26. Wilcoxon, F. Individual comparisons by ranking methods. In:*Breakthroughs in Statistics*; Springer: New York, NY, USA, 1992. [CrossRef]
27. Khan, M.; Naeem, M.R.; Al-Ammar, E.A.; Ko, W.; Vettikalladi, H.; Ahmad, I. Power forecasting of regional wind farms via variational auto-encoder and deep hybrid transfer learning. *Electronics* **2022**, *11*, 206. [CrossRef]
28. Van der Maaten, L.; Hinton, G. Visualizing data using t-SNE. *J. Mach. Learn. Res.* **2008**, *9*, 2579–2605.
29. Nivarthi, C.P.; Vogt, S.; Sick, B. Unified Autoencoder with Task Embeddings for Multi-Task Learning in Renewable Power Forecasting. In Proceedings of the 2022 21st IEEE International Conference on Machine Learning and Applications (ICMLA), Nassau, Bahamas, 12–15 December 2022; pp. 1530–1536. [CrossRef]