Enhancing Kubernetes Efficiency: Leveraging Artificial Neural Networks for Proactive Resource Management

Overview

Hypothesis: ANNs can predict Kubernetes load with high accuracy using historical monitoring data, thereby improving the efficiency of horizontal scaling decisions.

This proposal outlines a research project aimed at leveraging artificial neural networks (ANNs) to predict load for effective horizontal scaling in Kubernetes environments. The project seeks to enhance resource allocation, minimize costs, and improve system responsiveness.

Background

Kubernetes has become the leading platform for managing containerized applications, particularly in cloud environments. As Kubernetes applications scale, ensuring efficient resource allocation is crucial. This paper explores using ANNs to enhance resource scaling in Kubernetes clusters by predicting load and scaling needs based on historical data, reducing inefficiencies, and improving overall performance.

Problem Statement

Dynamic scaling in Kubernetes faces challenges due to the fluctuating loads and complex interactions within the cloud environment. Traditional methods that react to changes in load can lead to resource wastage or performance issues. By incorporating ANNs, Kubernetes can adopt a predictive scaling model that automates resource allocation based on anticipated demands. This method leverages the ability of ANNs to process large datasets and recognize patterns, enabling precise scaling that maintains performance and improves efficiency.

Dataset

Dataset Cite: Fernandes, Marcelo (2024), “Horizontal Scaling in Kubernetes Dataset Using Artificial Neural Networks for Load Forecasting”, Mendeley Data, V1, doi: 10.17632/ks9vbv5pb2.1

Reference: Horizontal Scaling in Kubernetes Dataset Using Artificial Neural Networks for Load Forecasting

This dataset provides a rich source of metrics collected under various traffic conditions to enhance horizontal scaling decisions in Kubernetes clusters through machine learning. The focus is on two primary types of data

Training Data (TrainData.csv):

To train the ANN models, providing them with a diverse set of scenarios to learn from. Includes aggregated metrics such as average CPU load, packet reception rates, and the number of pods. These metrics are averaged over multiple experiments, each conducted under different stress levels simulated by varying the number of virtual users and replicas. This structured approach helps in modeling the ANN to predict the optimal number of pods required to handle various loads efficiently.

Test Data: Split into two parts:

Kubernetes Cluster Data (TestK8sData.csv): Contains time-stamped data points capturing CPU usage, packet reception rates, the number of pods, and additional parameters like experiment type and CPU thresholds. This component is crucial for testing how well the ANN models can adapt to real-time data and actual load conditions.

JMeter Tool Data (TestJMeterData.csv): Provides logs from HTTP requests generated by JMeter, detailing metrics such as response times and success rates. This data helps assess the performance of the Kubernetes cluster in handling web traffic, crucial for evaluating the responsiveness of the ANN model.

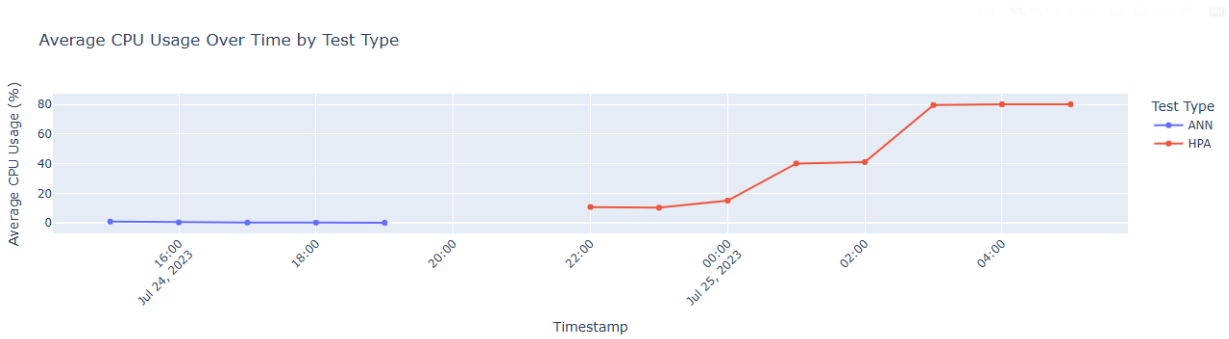
Methods

The project leverages Artificial Neural Networks (ANNs) for predictive scaling:

- Data Preprocessing: Cleaning and normalizing the dataset and derive new features to ensure the ANN model can efficiently learn from the data.
- Model Development: Using AWS SageMaker to develop and train the ANN models.
- Training & Validation: Split data into training and test sets, using the training set for model development and the test set to evaluate performance.
- Comparison: The performance of the ANN model will be compared with traditional horizontal pod autoscalers (HPA) in Kubernetes.

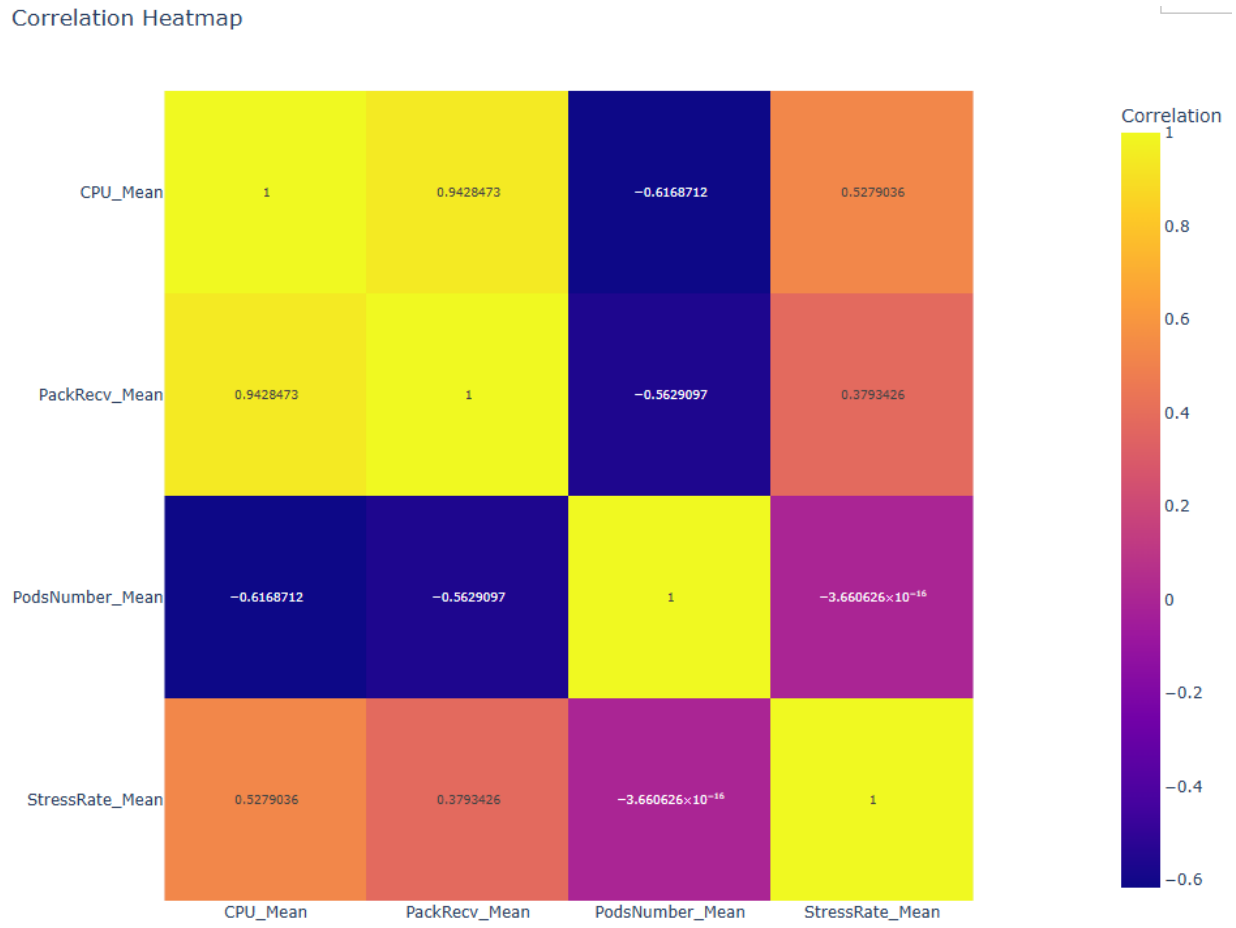
Exploratory Data Analysis

Combined EDA test_jmeter_data & test_k8s_data dataset



This visualization presents aggregated CPU usage over time, differentiated by test type (ANN vs. HPA) for Kubernetes clusters. By averaging CPU loads over hourly intervals, we can clearly see trends in resource utilization for each scaling strategy.an easy comparison of how each model adapts to changes in load over time, demonstrating the ANN model's responsiveness and its potential for optimizing resource allocation in dynamic environments.

Correlation Heatmap - Training Dataset



Key observations:

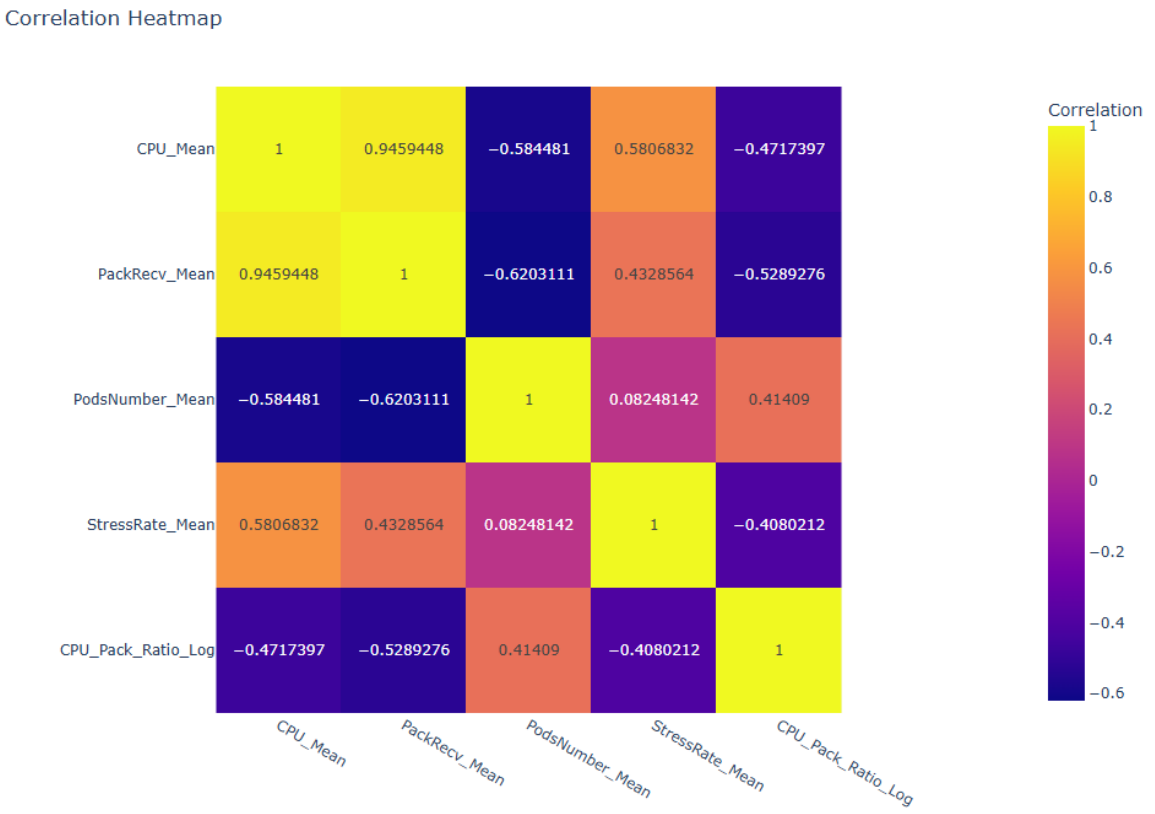
CPU_Mean and PackRecv_Mean have a strong positive correlation of 0.94. PodsNumber_Mean shows a negative correlation with CPU_Mean (-0.62) and PackRecv_Mean (-0.56). StressRate_Mean has moderate positive correlations with both CPU_Mean (0.53) and PackRecv_Mean (0.38). There is a negligible correlation between StressRate_Mean and PodsNumber_Mean (around 0).

Feature Engineering

A new feature, CPU_Pack_Ratio, is created by dividing CPU_Mean by PackRecv_Mean to capture the relationship between system load and network activity. To enhance model learning, a logarithmic

transformation is applied to this ratio, normalizing its distribution and stabilizing variance, making it more suitable for the model.

Correlation Matrix



CPU_Mean and PackRecv_Mean are highly positively correlated (0.95). This indicates that as CPU usage increases, the packet reception tends to increase as well.PodsNumber_Mean has a moderate negative correlation with both CPU_Mean (-0.58) and PackRecv_Mean (-0.62). This suggests that as CPU usage or packet reception increases, the number of pods tends to decrease.PodsNumber_Mean has a weak positive correlation with StressRate_Mean (0.08). This indicates that the stress rate might not have a strong impact on the number of pods.PodsNumber_Mean shows a weak positive correlation (0.41) with CPU_Pack_Ratio_Log, suggesting some relationship, though not very strong.

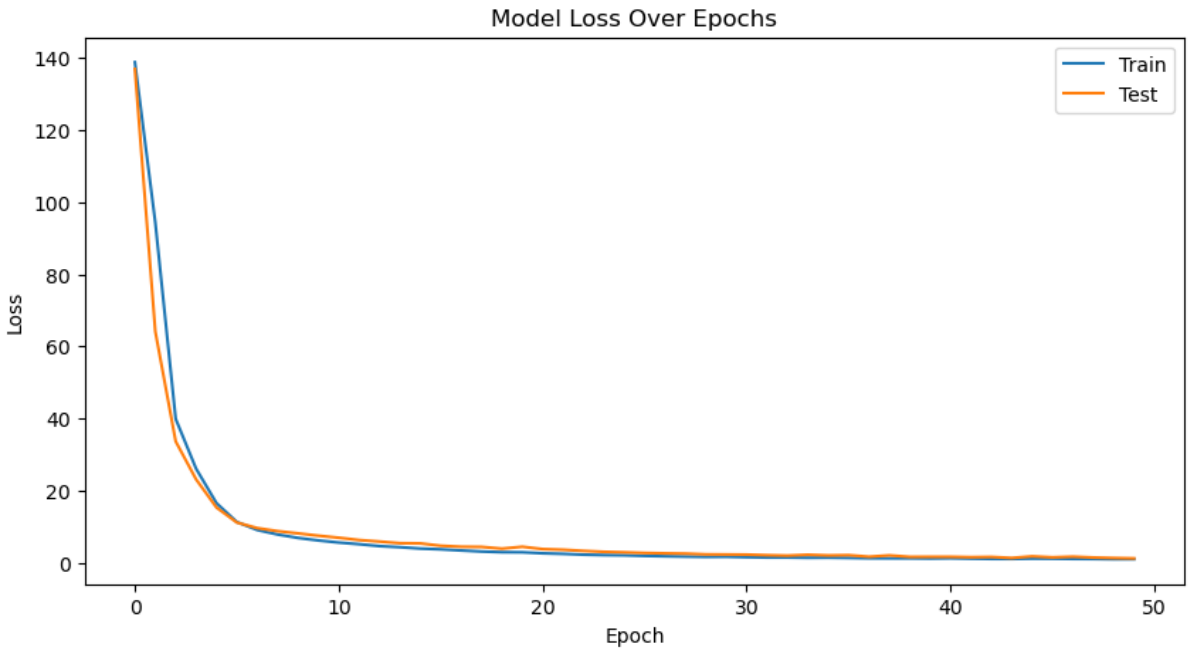
For predicting PodsNumber_Mean, the most important features appear to be CPU_Mean and PackRecv_Mean

Prediction Model Training

The model training process involved using a neural network with multiple hidden layers and a linear output layer to predict the required number of pods based on the input features. The model was trained over 50 epochs, with each epoch improving the model's ability to predict the target variable, as seen in the decreasing loss values. The performance of the model was evaluated using test data, and the final test loss reached 1.29, indicating that the model effectively learned from the data and made accurate predictions. This demonstrates that the neural network can provide reliable scaling decisions for Kubernetes clusters based on historical performance data.

Performance Evaluation

Model Loss Over Epochs



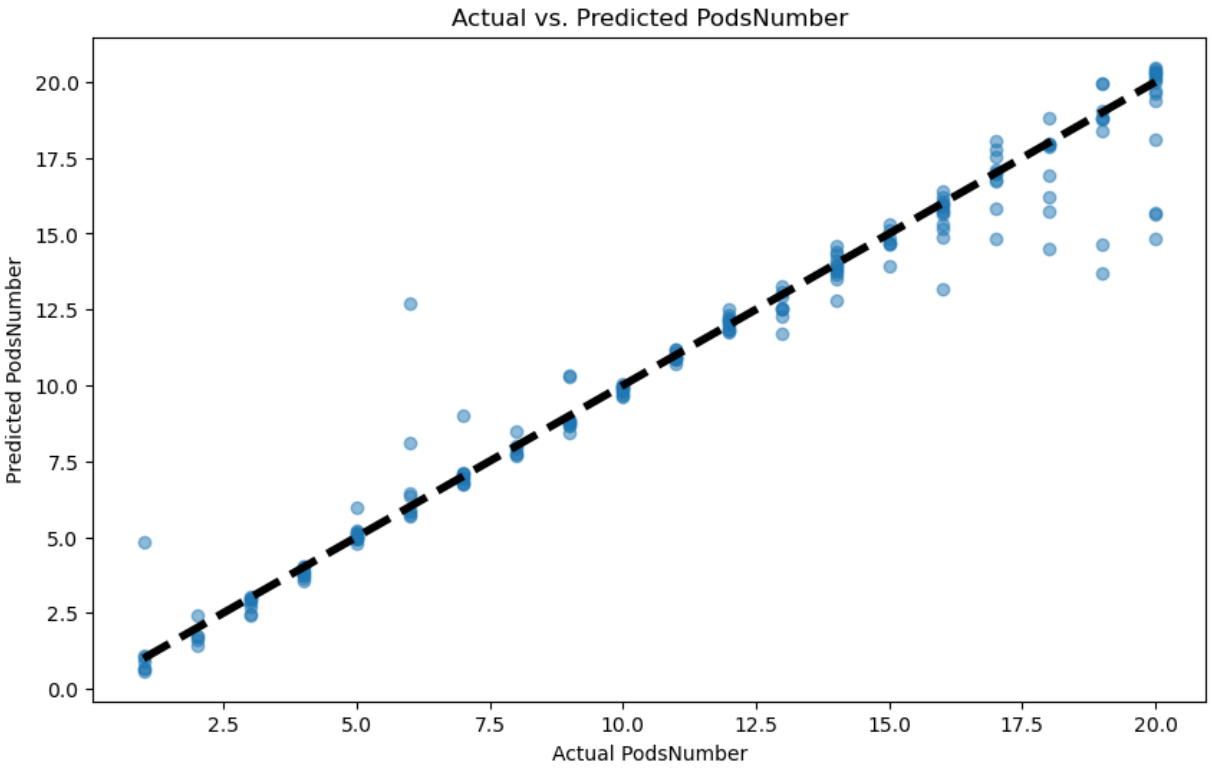
The plot of model loss over epochs shows how the training and test loss decrease as the model improves over time. Initially, there is a sharp drop in loss during the first few epochs, indicating that the model is quickly learning from the training data. As the training progresses, the loss levels off, signifying

that the model is converging and the learning rate is decreasing. This visualization demonstrates the model's ability to generalize, with the training and test loss curves remaining closely aligned.

Model Evaluation Metrics

After training, the model's performance was evaluated using key metrics, including Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), and R-squared (R^2). The RMSE is 1.14, indicating the average error magnitude, while the MAE of 0.54 shows the average absolute difference between predicted and actual values. The R^2 score of 0.96 indicates that the model explains 96% of the variance in the target variable, which suggests a highly accurate prediction.

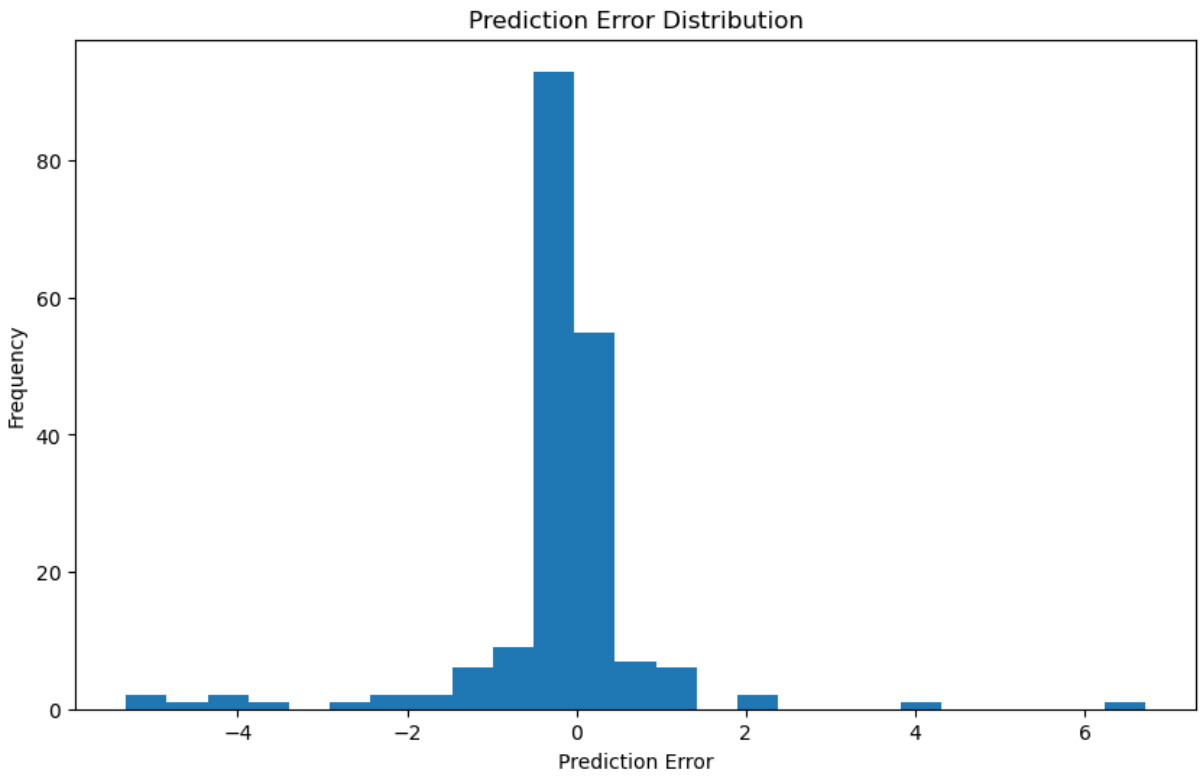
Actual vs Predicted PodNumbers



This scatter plot compares the actual values of the number of pods with the predicted values. The diagonal dashed line represents perfect predictions. Points are scattered around this line, with most predictions close to the actual values, demonstrating that the model can effectively predict the required

number of pods. A strong positive correlation is observed between the actual and predicted values, reinforcing the model's predictive accuracy.

Prediction Error Distribution



The distribution of prediction errors is shown in the histogram, where the majority of errors are close to zero. This suggests that the model's predictions are generally accurate, with few large deviations. The distribution's skewness to the right may indicate occasional under-predictions, but overall, the model performs well in minimizing large errors, which is crucial for real-time scaling decisions in Kubernetes environments.

Potential Ethical Implications

ANNs require large volumes of data to train effectively. In environments like Kubernetes, this data can include sensitive information about application usage, user behavior, and system performance. It's crucial to implement robust data governance policies to protect privacy and comply with regulations.

Over-reliance on automated systems can lead to a degradation of human expertise in managing cloud environments. Ensuring that human operators are kept in the loop and understand the basis of automated decisions is important to prevent over-dependence and maintain critical human oversight.

There is a risk that the models could develop biases based on the data they are trained on and The decision-making processes of ANNs can be opaque, often referred to as "black box" models. There must be a level of transparency about how predictive decisions are made.

Conclusion

ANNs show promise in predicting resource needs in Kubernetes environments, reducing resource wastage, and improving efficiency by providing accurate load forecasts. The approach is more proactive than traditional reactive scaling methods.

References

Dataset reference – Mendeley data, Fernandes, Marcelo (2024), “Horizontal Scaling in Kubernetes Dataset Using Artificial Neural Networks for Load Forecasting”, Mendeley Data, V1, doi: 10.17632/ks9vbv5pb2.1

Kubernetes Horizontal Pod Autoscaling, Horizontal Pod Autoscaling | Kubernetes

Machine Learning medium.com, Artificial Neural Network (ANN)