A. Training Data for Toy Regression Example

In this section we present the training samples used for evaluation of the toy regression example, presented in Figure 13. This data clearly shows how aleatoric uncertainty varies with the input (it increases linearly with x), which is denominated heteroscedatic uncertainty.

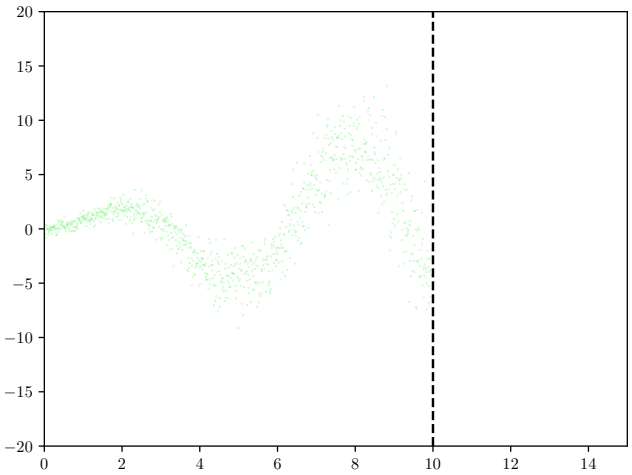

Figure 13. Training data used for the toy sinusoid problem with heteroscedatic aleatoric uncertainty.

B. Neural Network Architecture Details

1. **Regression task (Sinusoidal function with noise):**

   - General configurations for all networks: Number of epochs: 700. Batch size: 32. Optimizer: Adam ($lr = 0.001$, $\beta_1 = 0.9$, and $\beta_2 = 0.999$).

   - Configuration applied at each of the methods tested:

   - **Baseline**. Dense(32, ReLU) - Dense(32, ReLU).
   - **Dropout**. Dense(32, ReLU) - Dropout(0.25) - Dense(32, ReLU) - Dropout(0.25).
   - **DropConnect**. DropConnectDense(32, ReLU, $p = 0.1$) - DropConnectDense(32, ReLU, $p = 0.1$).
   - **Flipout**. FlipoutDense(32, ReLU) - FlipoutDense(32, ReLU). Prior is disabled.
   - **Ensembles**. 5 copies of the Baseline model trained with different random weight initializations.

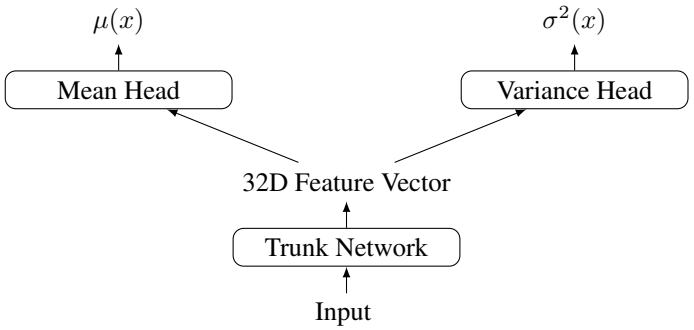

Figure 14. Diagram of the basic network architecture for regression with uncertainty. Each trunk network is a specific implementation of an uncertainty quantification method, as shown in the list above.

- Each of the networks displayed above are trunk networks. To predict regression values with uncertainty, two parallel layers are added. One Dense(1, Linear) for predicting the mean $\mu(x)$, and one Dense(1, Softplus) to predict the standard deviation $\sigma(x)$. This network configuration is visually displayed in Figure 14.

2. **Classification task (FER+ Dataset):**

   - General configurations for all networks: Number of epochs: 120. Batch size: 64. Optimizer: Adam ($lr = 0.001$, $\beta_1 = 0.9$, and $\beta_2 = 0.999$).

   - Configuration applied at each of the methods tested:

   - **Baseline**. Conv2D(64, 3×3, ReLU) - BatchNorm() - MaxPool2D(2×2) - Conv2D(128, 3×3, ReLU) - BatchNorm() - MaxPool2D(2×2) - Conv2D(128, 3×3, ReLU) - BatchNorm() - MaxPool2D(2×2) - Flatten() - Dense(256, ReLU) - Dense(256, ReLU).
   - **Dropout**. Conv2D(64, 3×3, ReLU) - BatchNorm() - MaxPool2D(2×2) - Conv2D(128, 3×3, ReLU) - BatchNorm() - MaxPool2D(2×2) - Conv2D(128, 3×3, ReLU) - BatchNorm() - MaxPool2D(2×2) - Flatten() - StochasticDropout($p = 0.25$) - Dense(256, ReLU) - StochasticDropout($p = 0.25$) - Dense(256, ReLU).
   - **DropConnect**. Conv2D(64, 3×3, ReLU) - BatchNorm() - MaxPool2D(2×2) - Conv2D(128, 3×3, ReLU) - BatchNorm() - MaxPool2D(2×2) - Conv2D(128, 3×3, ReLU) - BatchNorm() - MaxPool2D(2×2) - Flatten() - DropConnectDense(256, ReLU, $p = 0.10$) - DropConnectDense(256, ReLU, $p = 0.10$).
   - **Flipout**. Conv2D(64, 3×3, ReLU) - BatchNorm() - MaxPool2D(2×2) - Conv2D(128, 3×3, ReLU) - BatchNorm() - MaxPool2D(2×2) - Conv2D(128, 3×3, ReLU) - BatchNorm() - MaxPool2D(2×2) - Flatten() - FlipoutDense(256, ReLU, $p = 0.10$) - FlipoutDense(256, ReLU, $p = 0.10$).
   - **Ensembles**. 5 copies of the Baseline model trained with different random weight initializations.