Augmenting transferred representations for stock classification

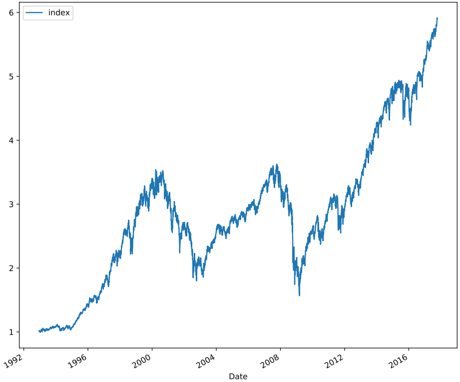
Elizabeth Fons[1] Paula Dawson[2] Xiao-jun Zeng[1] John Keane[1] Alexandros Iosifidis[3]

[1]University of Manchester [2]AllianceBernstein [3]Aarhus University

ICASSP 2021
11 June, 2021

Motivation

- Stock classification is challenging: high degree of noise and volatility due to external factors.
- One approach is to use a deep learning model to predict stock movement and then implement a trading rule to test profitability.
- In general, previous work has focused on predicting either movement of an index or of a small number of stocks.



Contribution

In this work, we present a model that can learn a trading rule directly from a large-scale stock dataset.

- We use transfer learning, where we pre-train a model with past returns of all constituent stocks of the S&P500 index, and then transfer it and fine-tune it on a dataset with the trading rule included.
- We propose the use of data augmentation on the feature space defined as the output of the pre-trained model and we compare this approach with the standard augmentation in the input space.
- We test our model by building the learned trading rule and calculate profitability taking into account transaction fees.

Problem formulation

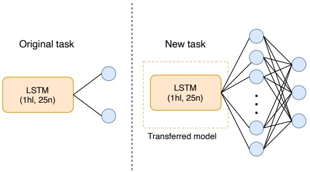
- Original task: Binary classification problem: 1 denotes up (price above the daily median) and 0 denotes down (price below the daily median).
- Trading rule: 3-class problem, top 10 stocks labelled *buy*, bottom 10 stocks labelled *sell* and the rest ($\approx$ 480) as *do nothing*.

- Daily returns of all S&P500 constituents between 1990 and 2018, divided into 25 data splits consisting of 750 days of training/validation and 250 days for testing.
- Each stock is segmented into sequences of 240 time steps used to predict the next day price movement ($(750 - 240) \cdot 500 \approx$255K).
- Cross-entropy loss with a term that maximizes returns:

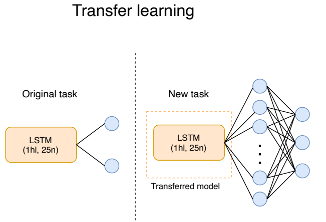
$$\mathcal{L}_{R+CE}(\Theta) = \mathcal{L}_{CE} + \alpha\mathcal{L}_{returns} = \mathcal{L}_{CE} + -\alpha\frac{1}{B}\sum R(i,t) \tag{1}$$

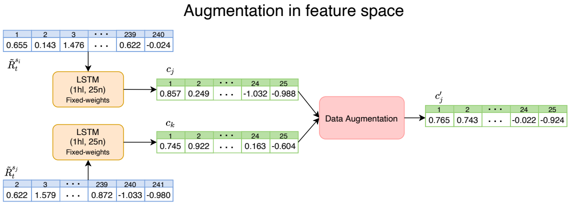
Training overview

Transfer learning

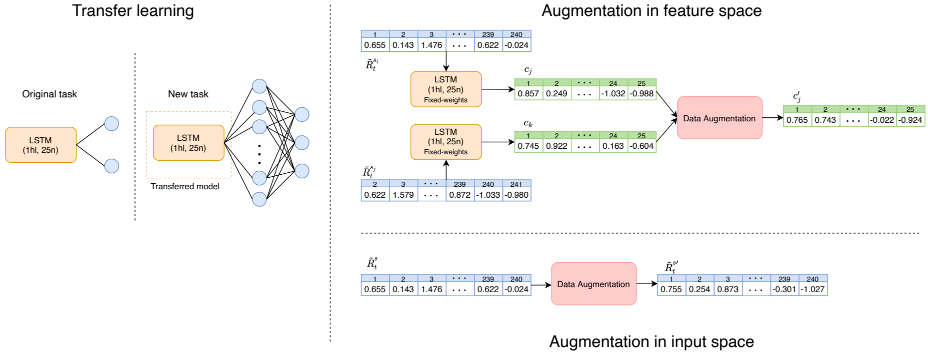


Training overview

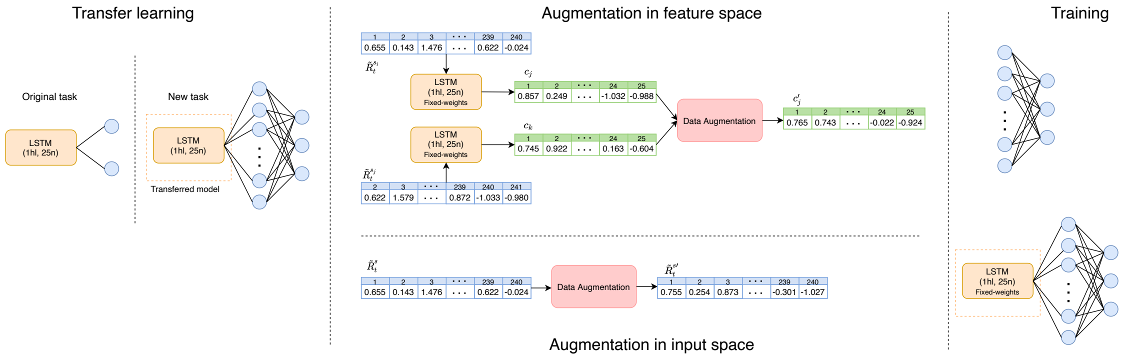




Training overview

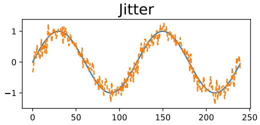


Training overview



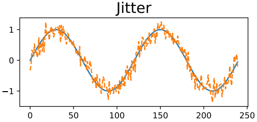
Data augmentation on input space

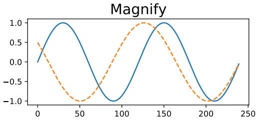
- **Jittering (Jit-inp):** Gaussian noise with a mean $\mu = 0$ and standard deviation $\sigma = 0.05$ is added to the time series.



Data augmentation on input space

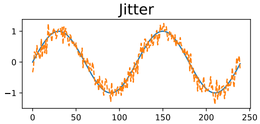
- **Jittering (Jit-inp):** Gaussian noise with a mean $\mu = 0$ and standard deviation $\sigma = 0.05$ is added to the time series.
- **Magnify:** a variation of window slicing, we randomly slice windows between 40% and 80% of the original time series, but always from the fixed end.

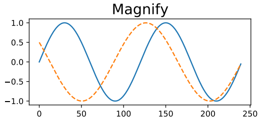


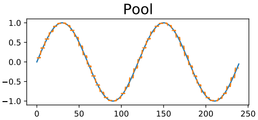


Data augmentation on input space

- **Jittering (Jit-inp):** Gaussian noise with a mean $\mu = 0$ and standard deviation $\sigma = 0.05$ is added to the time series.
- **Magnify:** a variation of window slicing, we randomly slice windows between 40% and 80% of the original time series, but always from the fixed end.
- **Pool:** Reduces the temporal resolution without changing the length of the time series by averaging a pooling window.

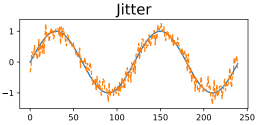


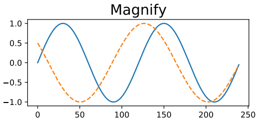


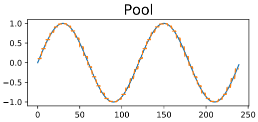


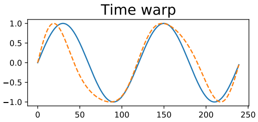
Data augmentation on input space

- **Jittering (Jit-inp):** Gaussian noise with a mean $\mu = 0$ and standard deviation $\sigma = 0.05$ is added to the time series.
- **Magnify:** a variation of window slicing, we randomly slice windows between 40% and 80% of the original time series, but always from the fixed end.
- **Pool:** Reduces the temporal resolution without changing the length of the time series by averaging a pooling window.
- **Time warp:** time intervals between samples are distorted based on a random smooth warping curve by cubic spline with four knots at random magnitude.



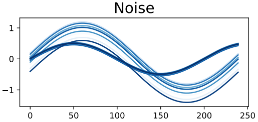






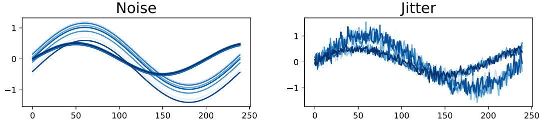
Data augmentation on feature space

- **Noise:** Gaussian noise is generated with zero mean and per-element standard deviation calculated across all transformed vectors; the noise is scaled by a global parameter γ:
  $c'_i = c_i + \gamma X, X \sim \mathcal{N}\{0, \sigma_i^2\}$



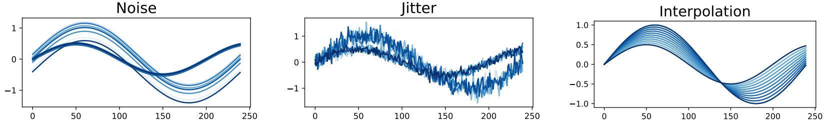
Data augmentation on feature space

- **Noise:** Gaussian noise is generated with zero mean and per-element standard deviation calculated across all transformed vectors; the noise is scaled by a global parameter γ:
  $\mathrm{c}'_i = \mathrm{c}_i + \gamma X, X \sim \mathcal{N}\{0, \sigma_i^2\}$
- **Jittering (Jit-feat):** Random noise with mean $\mu = 0$ and standard deviation $\sigma = 0.05$ is added to the context vector.



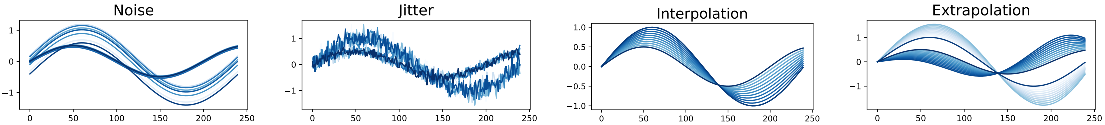
Data augmentation on feature space

- **Noise:** Gaussian noise is generated with zero mean and per-element standard deviation calculated across all transformed vectors; the noise is scaled by a global parameter γ:
  $c_i' = c_i + \gamma X, X \sim \mathcal{N}\{0, \sigma_i^2\}$
- **Jittering (Jit-feat):** Random noise with mean $\mu = 0$ and standard deviation $\sigma = 0.05$ is added to the context vector.
- **Interpolation:** for each sample, we find its K intra-class nearest neighbours in feature space and for each pair c_k and c_j, a new vector c_j' is generated using interpolation: $c_j' = (c_k - c_j)\lambda + c_j$



Data augmentation on feature space

- **Noise:** Gaussian noise is generated with zero mean and per-element standard deviation calculated across all transformed vectors; the noise is scaled by a global parameter γ:
  $c'_i = c_i + \gamma X, X \sim \mathcal{N}\{0, \sigma_i^2\}$
- **Jittering (Jit-feat):** Random noise with mean $\mu = 0$ and standard deviation $\sigma = 0.05$ is added to the context vector.
- **Interpolation:** for each sample, we find its K intra-class nearest neighbours in feature space and for each pair c_k and c_j, a new vector c'_j is generated using interpolation: $c'_j = (c_k - c_j)\lambda + c_j$
- **Extrapolation:** similarly, we apply extrapolation to the feature space vectors in the following way, with $\lambda = 0.2$: $c'_j = (c_j - c_k)\lambda + c_j$



Results

Table: Performance of the $k = 10$ long-short portfolios after transaction costs, for the TL+FC(25) model trained with different augmentation methods and the combined loss $\mathcal{L}_{R+CE}$.

Method	Ann ret	Ann vol	IR	D. Risk	DIR	Acc	Macro-F1
LSTM	29.2	**28.66**	1.02	19.08	1.53	—	—
No TL (25)+$\mathcal{L}_{CE}$	12.99	38.15	0.34	25.48	0.52	73.13±18.94	33.57 ± 6.5
TL+FC(25)+$\mathcal{L}_{CE}$	32.25	30.29	1.06	19.6	1.65	68.34±16.5	31.79±5.12
TL+FC(25)+$\mathcal{L}_{R+CE}$	34.62	30.20	1.15	19.59	1.77	64.79±16.86	30.72±5.28

Results

Table: Performance of the $k = 10$ long-short portfolios after transaction costs, for the TL+FC(25) model trained with different augmentation methods and the combined loss $\mathcal{L}_{R+CE}$.

Method	Ann ret	Ann vol	IR	D. Risk	DIR	Acc	Macro-F1
LSTM	29.2	**28.66**	1.02	19.08	1.53	—	—
No TL (25)+$\mathcal{L}_{CE}$	12.99	38.15	0.34	25.48	0.52	73.13±18.94	33.57 ± 6.5
TL+FC(25)+$\mathcal{L}_{CE}$	32.25	30.29	1.06	19.6	1.65	68.34±16.5	31.79±5.12
TL+FC(25)+$\mathcal{L}_{R+CE}$	34.62	30.20	1.15	19.59	1.77	64.79±16.86	30.72±5.28
TL+FC(25) Extrapolation	**39.70**	29.43	**1.35**	18.96	**2.09**	62.90±17.87	30.10±5.81
TL+FC(25) Interpolation	36.87	29.69	1.24	**18.93**	1.95	62.46±17.80	29.95±5.74
TL+FC(25) Noise	30.97	29.15	1.06	19.14	1.62	62.43±18.12	29.95±5.81
TL+FC(25) Jitter-feat	39.11	29.93	1.31	19.22	2.03	62.71±17.84	30.04±5.71

Results

Table: Performance of the $k = 10$ long-short portfolios after transaction costs, for the TL+FC(25) model trained with different augmentation methods and the combined loss $\mathcal{L}_{R+CE}$.

Method	Ann ret	Ann vol	IR	D. Risk	DIR	Acc	Macro-F1
LSTM	29.2	**28.66**	1.02	19.08	1.53	—	—
No TL (25)+$\mathcal{L}_{CE}$	12.99	38.15	0.34	25.48	0.52	73.13±18.94	33.57 ± 6.5
TL+FC(25)+$\mathcal{L}_{CE}$	32.25	30.29	1.06	19.6	1.65	68.34±16.5	31.79±5.12
TL+FC(25)+$\mathcal{L}_{R+CE}$	34.62	30.20	1.15	19.59	1.77	64.79±16.86	30.72±5.28
TL+FC(25) Extrapolation	**39.70**	29.43	**1.35**	18.96	**2.09**	62.90±17.87	30.10±5.81
TL+FC(25) Interpolation	36.87	29.69	1.24	**18.93**	1.95	62.46±17.80	29.95±5.74
TL+FC(25) Noise	30.97	29.15	1.06	19.14	1.62	62.43±18.12	29.95±5.81
TL+FC(25) Jitter-feat	39.11	29.93	1.31	19.22	2.03	62.71±17.84	30.04±5.71
TL+FC(25) Jitter-input	29.74	39.94	0.96	20.12	1.48	68.23±16.62	31.75±5.06
TL+FC(25) Magnify	20.39	29.41	0.69	19.86	1.03	63.78±16.78	30.42±5.47
TL+FC(25) Pool	27.18	29.96	0.91	19.64	1.38	57.71±17.43	28.38±5.72
TL+FC(25) Time Warp	32.76	29.46	1.11	19.21	1.71	61.81±19.96	29.80±5.48

Results

Table: Performance of the $k = 10$ long-short portfolios after transaction costs, for the TL+FC(100) model trained with different augmentation methods and the combined loss $\mathcal{L}_{R+CE}$.

Method	Ann ret	Ann vol	IR	D. Risk	DIR	Acc	Macro-F1
LSTM	29.2	28.66	1.02	19.08	1.53	—	—
No TL (100)+$\mathcal{L}_{R+CE}$	21.05	39.95	0.55	25.9	0.84	57.02±21.95	28.24±8.12
TL+FC(100)+$\mathcal{L}_{CE}$	30.83	30.31	1.02	19.79	1.56	68.88±15.93	31.95±4.76
TL+FC(100)+$\mathcal{L}_{R+CE}$	32.14	29.97	1.07	19.87	1.62	64.72±17.25	30.7±5.41

Results

Table: Performance of the $k = 10$ long-short portfolios after transaction costs, for the TL+FC(100) model trained with different augmentation methods and the combined loss $\mathcal{L}_{R+CE}$.

Method	Ann ret	Ann vol	IR	D. Risk	DIR	Acc	Macro-F1
LSTM	29.2	28.66	1.02	19.08	1.53	—	—
No TL (100)+$\mathcal{L}_{R+CE}$	21.05	39.95	0.55	25.9	0.84	57.02±21.95	28.24±8.12
TL+FC(100)+$\mathcal{L}_{CE}$	30.83	30.31	1.02	19.79	1.56	68.88±15.93	31.95±4.76
TL+FC(100)+$\mathcal{L}_{R+CE}$	32.14	29.97	1.07	19.87	1.62	64.72±17.25	30.7±5.41
TL+FC(100) Extrapolation	27.38	29.33	0.93	19.42	1.41	62.74±17.73	30.05±5.81
TL+FC(100) Interpolation	30.84	29.72	1.04	19.38	1.59	62.49±17.35	30.01±5.63
TL+FC(100) Noise	29.02	29.44	0.99	19.2	1.51	62.2±17.82	29.87±5.73
TL+FC(100) Jitter-feat	**37.14**	29.31	**1.27**	**18.92**	**1.96**	61.84±17.89	29.75±5.75

Results

Table: Performance of the $k = 10$ long-short portfolios after transaction costs, for the TL+FC(100) model trained with different augmentation methods and the combined loss $\mathcal{L}_{R+CE}$.

Method	Ann ret	Ann vol	IR	D. Risk	DIR	Acc	Macro-F1
LSTM	29.2	28.66	1.02	19.08	1.53	—	—
No TL (100)+$\mathcal{L}_{R+CE}$	21.05	39.95	0.55	25.9	0.84	57.02±21.95	28.24±8.12
TL+FC(100)+$\mathcal{L}_{CE}$	30.83	30.31	1.02	19.79	1.56	68.88±15.93	31.95±4.76
TL+FC(100)+$\mathcal{L}_{R+CE}$	32.14	29.97	1.07	19.87	1.62	64.72±17.25	30.7±5.41
TL+FC(100) Extrapolation	27.38	29.33	0.93	19.42	1.41	62.74±17.73	30.05±5.81
TL+FC(100) Interpolation	30.84	29.72	1.04	19.38	1.59	62.49±17.35	30.01±5.63
TL+FC(100) Noise	29.02	29.44	0.99	19.2	1.51	62.2±17.82	29.87±5.73
TL+FC(100) Jitter-feat	**37.14**	29.31	**1.27**	**18.92**	**1.96**	61.84±17.89	29.75±5.75
TL+FC(100) Jitter-input	29.49	30.32	0.97	19.73	1.49	67.79±17.18	31.64±5.46
TL+FC(100) Magnify	22.11	30.36	0.73	20.21	1.09	67.12±16.68	30.34±5.27
TL+FC(100) Pool	27.64	29.50	0.94	18.98	1.46	57.64±17.89	28.37±5.89
TL+FC(100) Time Warp	26.64	29.65	0.90	19.49	1.37	65.55±18.03	29.69±5.89

Conclusions

- Using transfer learning on a stock classification task where a trading rule is included in the training dataset improves financial performance when compared to training a neural network from scratch.
- Using a training loss that combines a classification objective with maximization of returns improves risk adjusted returns when compared with the single cross-entropy loss.
- We investigated the use of data augmentation on the feature space (defined as the output of the pre-trained model) and compared it with traditional data augmentation methods on the input space. Augmentation on the feature space improves up to 20% risk adjusted returns when compared to a transferred model without augmentation.

Thank you!