AI TOOLS FOR ACTUARIES
Chapter 5: FNN Example - CANN Approach

Ronald Richman, Salvatore Scognamiglio, Mario V. Wüthrich

2025-09-15

Boosting a GLM with a neural network

Overview

- This notebook presents the idea of applying a FNN boosting step to a GLM to enhance the GLM regression structure with FNN features.
- This approach may also help one to improve the GLM structure, as it can identify missing structure in GLMs.

CANN and ResNet

- The idea that we are going to present has been discussed in an editorial of the ASTIN Bulletin called ‘Yes, we CANN!’ to promote machine learning research within the actuarial community; see Wüthrich and Merz (2019).
    - CANN = Combined Actuarial and Neural Network
- The CANN proposal is rather similar to the ResNet (residual network) idea of He *et al.* (2015).
- For boosting, in general, and gradient boosting, in particular, we also refer to our chapter on gradient boosting machines (GBMs).

Boosting idea

- We start from a MLE fitted GLM
$$\widehat{\mu}^{\mathrm{GLM}}(\boldsymbol{X}) := \mu^{\mathrm{GLM}}_{\widehat{\vartheta}^{\mathrm{MLE}}}(\boldsymbol{X}) = g^{-1}\left\langle \widehat{\vartheta}^{\mathrm{MLE}}, \boldsymbol{X} \right\rangle .$$
- This step used the learning sample $\mathcal{L} = (Y_i, \boldsymbol{X}_i, v_i)_{i=1}^n$, giving us the *estimated residuals*
$$\widehat{\varepsilon}_i = Y_i - \widehat{\mu}^{\mathrm{GLM}}(\boldsymbol{X}_i).$$
- These estimated residuals do not contain systematic structure if the fitted GLM is suitable.
- This motivates a second regression step on the new learning sample $\mathcal{L}^{\varepsilon} = (\widehat{\varepsilon}_i, \boldsymbol{X}_i, v_i)_{i=1}^n$:
    - The idea of *boosting* is to iteratively improve the model by focusing on finding the weaknesses of the previous model(s).
    - This is quite different from *ensembling*, which (only) averages over the models, but it does not let them compete to improve the prediction.

Excursion: estimated residuals and dispersion

Tip

The fast reader can skip this section and come back to it later.

- For the Poisson model, we have dispersion parameter $\varphi = 1$.
- Dispersion estimation only becomes important in other EDF models, if one wants to assess higher moments.

Excursion: estimated residuals

The residuals $\widehat{\varepsilon}_i$ defined above need some care:

- They are not independent, even if the instances $1 \leq i \leq n$ are independent. The learning sample $\mathcal{L}$ enters the estimated regression function $\widehat{\mu}^{\mathrm{GLM}} = \mu^{\mathrm{GLM}}_{\widehat{\vartheta}^{\mathrm{MLE}}}$; supposed we consider a GLM. Generally, this makes the estimated residuals correlated.
- This is the reason for normalizing the empirical variance estimator by $1/(n-1)$ and not by $1/n$ (to receive unbiasedness).
- Generally, the above residuals have different variances, i.e., they are not i.i.d. (but only centered).
- Working within the EDF with cumulant function κ, the correct standardization is straightforward by using the variance function $\mu \mapsto V(\mu) := (\kappa'' \circ h)(\mu)$ with canonical link h; see EDF section.
- This motivates Pearson's residuals defined next.

Excursion: Pearson's residuals

- *Pearson's residuals* are defined by

$$\widehat{\varepsilon}_i^P = \frac{Y_i - \widehat{\mu}^{\mathrm{GLM}}(\boldsymbol{X}_i)}{\sqrt{V\left(\widehat{\mu}^{\mathrm{GLM}}(\boldsymbol{X}_i)\right)/v_i}}.$$

- Pearson's residuals should roughly look like an independent sequence of centered variables having the same dispersion, in particular, they should not show any systematic structure in the covariates.
- If Pearson's residuals show some systematic structure (beyond their mean parameters), either the estimated regression function is not correctly specified – apply a boosting step! – or the selected cumulant function κ does not provide the correct variance structure.
- We refer to Delong and Wüthrich (2025) for variance back-testing by using an isotonic regression.

Excursion: dispersion estimate

- Pearson's residuals motivate *Pearson's dispersion estimate*

$$\widehat{\varphi}^P = \frac{1}{n-(q+1)} \sum_{i=1}^{n} \left(\widehat{\varepsilon}_i^P\right)^2,$$

  if the estimated GLM function $\widehat{\mu}^{\mathrm{GLM}}$ involves $q+1$ estimated regression parameters $\widehat{\vartheta}^{\mathrm{MLE}} \in \mathbb{R}^{q+1}$.

- Besides Pearson's dispersion estimate, there is also the *deviance dispersion estimate* (not explicitly shown here, but used in the Tukey–Anscombe plot).

- In general, Pearson's dispersion estimate is consistent (in the statistical sense), whereas the deviance dispersion estimate is not.

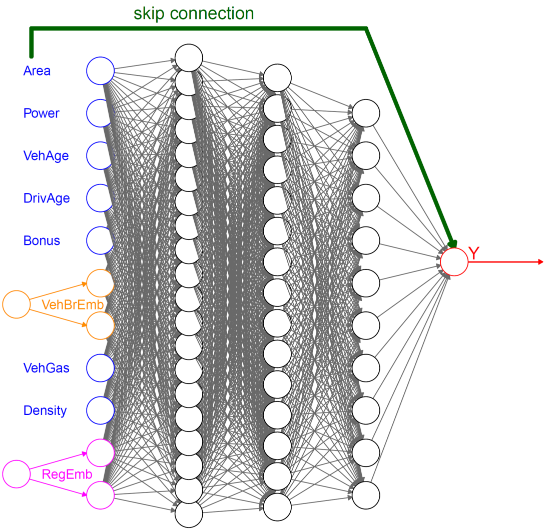
CANN proposal

- The CANN proposal of Wüthrich and Merz (2019) does not explicitly compute the residuals, but it directly acts on the regression function of the first estimated model, i.e., it acts on the linear predictor scale.
- Based on $\widehat{\mu}^{\rm GLM}$ with link g, we make the regression Ansatz

$$\begin{aligned}\mu^{\rm CANN}(\boldsymbol{X}) &= g^{-1}\left(g\left(\widehat{\mu}^{\rm GLM}(\boldsymbol{X})\right)+\left\langle \boldsymbol{w}^{(d+1)}, \boldsymbol{z}^{(d:1)}(\boldsymbol{X})\right\rangle\right)\\ &= g^{-1}\left(\left\langle \widehat{\vartheta}^{\rm MLE}, \boldsymbol{X}\right\rangle+\left\langle \boldsymbol{w}^{(d+1)}, \boldsymbol{z}^{(d:1)}(\boldsymbol{X})\right\rangle\right).\end{aligned}$$

- The MLE $\widehat{\vartheta}^{\rm MLE}$ has been fitted in a first GLM step and is kept fixed (frozen) in the (second) boosting step (which only fits the FNN part).
- If the fitted FNN part of this second step is equal to zero, we are back in the GLM, i.e., there has no additional structure been found.

- Keeping the first part $\langle \widehat{\vartheta}^{\mathrm{MLE}}, \boldsymbol{X} \rangle$ frozen during the second boosting step can be interpreted as having an *offset* playing the role of *known prior differences*.
- For fitting the FNN we apply the SGD algorithm on training and validation data as explained above.
- This two step estimation concept has been applied successfully in many studies, see, e.g., Brauer (2024) and Havrylenko and Heger (2024), the latter reference uses this CANN approach to detect interactions missing in GLMs; see also Example 5.11 in Wüthrich and Buser (2016).
- The similarity to ResNet of He *et al.* (2015) is apparent. The second term describes a classic FNN architecture, the first term can be interpreted as a *residual connection* or a *skip connection* because it connects the input $\boldsymbol{X}$ directly to the output (skipping the FNN); we illustrate this with Figure 7.14 of Wüthrich and Merz (2023).


skip connection
Area
Power
VehAge
DrivAge
Bonus
VehBrEmb
VehGas
Density
RegEmb
Y

CANN implementation

- We illustrate the CANN proposal on the French MTPL claims frequency data.
- We start with the Poisson GLM example from the GLM chapter, and in the second step we add the FNN features to this GLM.
- Load the French MTPL data:

```
load(file="../Data/freMTPL2freqClean.rda")
dat <- freMTPL2freqClean
```

Pre-process data for the GLM

- We use the same GLM as in the GLM notebook.

```
# this is identical to the GLM notebook
dat$AreaGLM       <- as.integer(dat$Area)
dat$VehPowerGLM   <- as.factor(pmin(dat$VehPower, 9))
dat$VehAgeGLM     <- as.factor(cut(dat$VehAge, c(0, 5, 12, 101),
                       labels = c("0-5", "6-12", "12+"),
                       include.lowest = TRUE))
dat$DrivAgeGLM    <- as.factor(cut(dat$DrivAge, c(18, 20, 25, 30, 40, 50, 70,
↪   101),
                       labels = c("18-20", "21-25", "26-30", "31-40", "41-50",
                       ↪   "51-70", "71+"),
                       include.lowest = TRUE))
dat$DrivAgeGLM    <- relevel(dat[,"DrivAgeGLM"], ref="31-40")
dat$BonusMalusGLM <- pmin(dat$BonusMalus, 150)
dat$DensityGLM    <- log(dat$Density)
```

Standardization function (continuous covariates FNN)

- The FNN pre-processing is identical to the FNN notebook with entity embeddings.

```
# standardization of continuous covariates
PreProcess.Continuous <- function(var1, dat2){
   names(dat2)[names(dat2) == var1]  <- "V1"
   dat2$X <- as.numeric(dat2$V1)
   dat2$X <- (dat2$X-mean(dat2$X))/sd(dat2$X)
   names(dat2)[names(dat2) == "V1"]  <- var1
   names(dat2)[names(dat2) == "X"]   <- paste(var1,"X", sep="")
   dat2
   }
```

- Again, the learning $\mathcal{L}$ and the test $\mathcal{T}$ samples require the identical scaling, and the scaling constants should be stored for pre-processing new data.

Covariate pre-processing for FNN (with entity embedding)

```
Features.PreProcess <- function(dat2){
   dat2 <- PreProcess.Continuous("Area", dat2)
   dat2 <- PreProcess.Continuous("VehPower", dat2)
   dat2$VehAge <- pmin(dat2$VehAge, 20)
   dat2 <- PreProcess.Continuous("VehAge", dat2)
   dat2$DrivAge <- pmin(dat2$DrivAge, 90)
   dat2 <- PreProcess.Continuous("DrivAge", dat2)
   dat2$BonusMalus <- pmin(dat2$BonusMalus, 150)
   dat2 <- PreProcess.Continuous("BonusMalus", dat2)
   dat2$VehBrandX <- as.integer(dat2$VehBrand) - 1      # tokenization embedding
   dat2$VehGasX <- as.integer(dat2$VehGas) - 1
   dat2$Density <- round(log(dat2$Density), 2)
   dat2 <- PreProcess.Continuous("Density", dat2)
   dat2$RegionX <- as.integer(dat2$Region) - 1          # tokenization embedding
   dat2
   }
```

Constructing learning and test samples

- Load TensorFlow and Keras:

```
library(tensorflow)
library(keras)
#library(keras3)        # this notebook uses Keras 2
```

- Covariate pre-processing and learning and test samples building.

```
## FNN covariate pre-processing
dat <- Features.PreProcess(dat)

## learning and test sample partition
learn <- dat[which(dat$LearnTest=='L'),]
test  <- dat[which(dat$LearnTest=='T'),]
```

Fit Poisson log-link GLM (limited covariates)

```
d.glm <- glm(ClaimNb ~ DrivAgeGLM + VehBrand + VehGas + DensityGLM + AreaGLM,
↪  data=learn, offset=log(Exposure), family=poisson())
```

- Poisson deviance loss (in-sample and out-of-sample):

```
Poisson.Deviance <- function(pred, obs, weight){
  10^2 * 2*(sum(pred)-sum(obs)+sum(log((obs/pred)^(obs))))/sum(weight)}

#
learn$GLM <- fitted(d.glm)
test$GLM  <- predict(d.glm, newdata=test, type="response")

#
(loss.GLM <- round(c(Poisson.Deviance(learn$GLM, learn$ClaimNb, learn$Exposure),
↪  Poisson.Deviance(test$GLM, test$ClaimNb, test$Exposure)), 3))
```

```
[1] 46.954 47.179
```

Prepare data for the CANN step

```
# covariate selection
features <- c("AreaX", "VehPowerX", "VehAgeX", "DrivAgeX", "BonusMalusX",
↪  "VehGasX", "DensityX", "VehBrandX", "RegionX")
# learning and test samples
Xlearn <- as.matrix(learn[, features])  # design matrix learning sample
Xtest  <- as.matrix(test[, features])   # design matrix test sample
Ylearn <- as.matrix(learn$ClaimNb)      # response learning sample
Ytest  <- as.matrix(test$ClaimNb)       # response test sample

# offset for the CANN step: we replace the volumes by the GLM prediction
GLMlearn <- as.matrix(learn$GLM)        # GLM prediction
GLMtest  <- as.matrix(test$GLM)         # GLM prediction
```

- In the remainder we apply precisely the same model as in the FNN notebook (with entity embedding).

FNN architecture of depth 3 (with entity embedding)

```
FNN.embed <- function(seed, q0, kk, bb){  # this uses Keras 2
    k_clear_session(); set.seed(seed); set_random_seed(seed)
    Design   <- layer_input(shape = c(q0[1]), dtype = 'float32')
    Volume   <- layer_input(shape = c(1), dtype = 'float32')
    BrandEmb = Design[,q0[1]-1] %>% layer_embedding(input_dim = kk[1], output_dim
↪   = bb[1], input_length = 1) %>% layer_flatten()
    RegionEmb = Design[,q0[1]] %>% layer_embedding(input_dim = kk[2], output_dim
↪   = bb[2], input_length = 1) %>% layer_flatten()
    #
    Network = list(Design[,1:(q0[1]-2)], BrandEmb, RegionEmb) %>%
        layer_concatenate() %>%
        layer_dense(units=q0[2], activation='tanh') %>%
        layer_dense(units=q0[3], activation='tanh') %>%
        layer_dense(units=q0[4], activation='tanh', name="FE") %>%
        layer_dense(units=1, activation='exponential')
    Response = list(Network, Volume) %>% layer_multiply()
    keras_model(inputs = c(Design, Volume), outputs = c(Response))
  }
```

Define and initialize FNN architecture

```
# define the network architecture
q00   <- c(length(features), c(20,15,10))    # FNN architecture
seed  <- 100
kk    <- c(length(unique(learn$VehBrandX)), length(unique(learn$RegionX)))
bb    <- c(2,2)

model <- FNN.embed(seed, q00, kk, bb)      # define network

# initialize for the GLM (the offset corresponds to the GLM and we ensure that
↪  there is no initial signal from the FNN part)
w0       <- get_weights(model)
w0[[9]]  <- array(0, dim=dim(w0[[9]]))
w0[[10]] <- array(0, dim=dim(w0[[10]]))
set_weights(model, w0)
```

- The network initialization differs from the classical FNN case because we already have the GLM parameters in the model from the first fitting step. Therefore, we initialize the read-out of the FNN to zero.

GLM: Poisson deviance losses

```
## GLM case not considering the FNN part (due to the above initialization)
learn.GLM <- model %>% predict(list(Xlearn, GLMlearn), batch_size=10^6)
test.GLM  <- model %>% predict(list(Xtest, GLMtest), batch_size=10^6)
#
loss.GLM2 <- round(c(Poisson.Deviance(learn.GLM, Ylearn, learn$Exposure),
                     Poisson.Deviance(test.GLM, Ytest, test$Exposure)), 3)
```

- In-sample and out-of-sample Poisson deviance losses (GLM):

```
# check for correct initialization
rbind(loss.GLM, loss.GLM2)
```

```
            [,1]   [,2]
loss.GLM  46.954 47.179
loss.GLM2 46.954 47.179
```

- Thus, we have correctly initialized.

Stochastic gradient descent fitting

```
## define callback for early stopping
if (!dir.exists("./Networks")){dir.create("./Networks")}
path1 <- paste0("./Networks/FNN1_CANN",seed,".h5")
CBs <- callback_model_checkpoint(path1, monitor = "val_loss", verbose = 0,
↪  save_best_only = TRUE, save_weights_only = TRUE)

#
model %>% compile(loss = 'poisson', optimizer = 'nadam')

#
fit <- model %>% fit(list(Xlearn, GLMlearn),  Ylearn,
                     validation_split = 0.1, batch_size = 5000, epochs = 200,
                     ↪  verbose = 0, callbacks = CBs)
#
which.min(fit[[2]]$val_loss)
```

```
[1] 90
```

FNN architecture: Poisson deviance loss results

```
# get optimal CANN weights
load_model_weights_hdf5(model, path1)
#
learn$CANN <- model %>% predict(list(Xlearn, GLMlearn), batch_size=10^6)
test$CANN  <- model %>% predict(list(Xtest, GLMtest), batch_size=10^6)
```

- In-sample and out-of-sample Poisson deviance losses: GLM and CANN

```
rbind(loss.GLM, loss.CANN <- round(c(Poisson.Deviance(learn$CANN, Ylearn,
↪  learn$Exposure), Poisson.Deviance(test$CANN, Ytest, test$Exposure)), 3))
```

```
            [,1]   [,2]
loss.GLM  46.954 47.179
loss.CANN 44.943 44.956
```

- Thus, the initial GLM has significantly been improved.

Summary CANN results

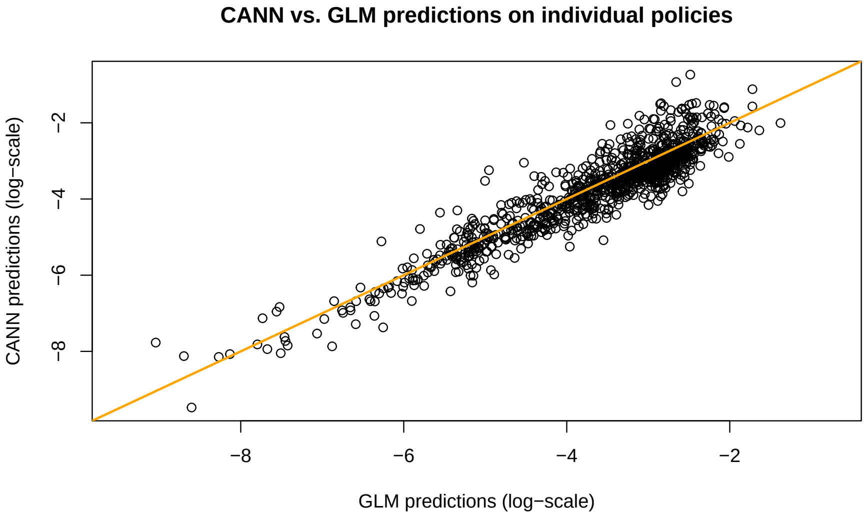
- In this example, the CANN approach leads to a clear improvement.
- This is not surprising, because the GLM did not consider all covariates (to speed up the computation time).
- Whenever the functional forms or the interactions are not correctly specified in the GLM, the FNN will boost the first GLM results.
- To understand whether the marginal functional forms or the interactions are not properly captured by the GLM, one could replace the GLM by a generalized additive model (GAM) in the first step. A GAM is flexible in modeling marginals, but it also more computational. If a CANN boosting step on the GAM leads to major improvements, they indicate missing interactions in the GAM. This has been considered in Example 5.11 of Wüthrich and Buser (2016).

Comparison to plain-vanilla FNNs

model	*in-sample*	*out-of-sample*	*balance (in %)*
Poisson null model	47.722	47.967	7.36
Poisson GLM	46.954	47.179	7.36
Poisson FNN	44.846	44.925	7.17
embedding FNN	45.030	44.948	7.16
CANN	44.943	44.956	7.17

- The Poisson GLM does not consider all covariates in this example.
- The standard deviations in in-sample and out-of-sample losses were 0.053 and 0.034, respectively (see notebook FNN: one-hot encoding).
- Often, the CANN performs slightly worse than a plain-vanilla FNN, but it gives more insight into the regression structure.
- We refrain from applying the balance correction step; this can be done analogously to the previous notebooks.

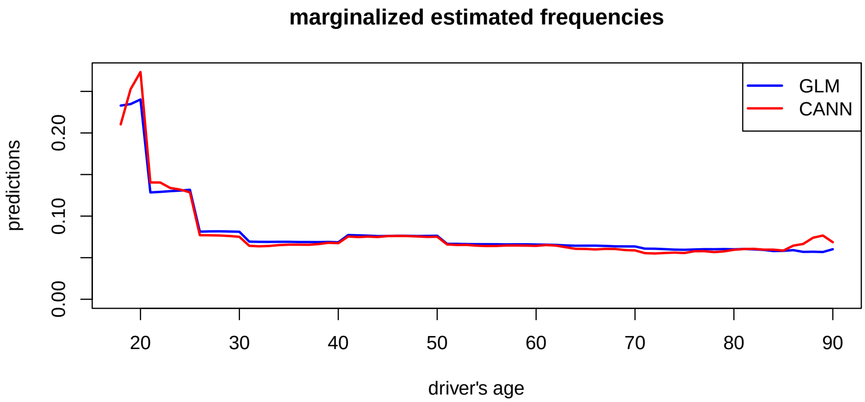
Illustration CANN vs. GLM predictions



Marginalized estimated frequencies

```
library(arrow)
library(tidyverse)
# compute marginalized frequencies
aggr <- data.frame(learn %>% group_by(DrivAge) %>%
                          summarize(GLM      = sum(GLM),
                                    CANN     = sum(CANN),
                                    Exposure = sum(Exposure)))
aggr$GLM <- aggr$GLM/aggr$Exposure
aggr$CANN <- aggr$CANN/aggr$Exposure

# plot marginalized frequencies
plot(x=aggr$DrivAge, y=aggr$GLM, ylim=range(aggr[,c("GLM","CANN")],0), type='l',
↪  lwd=2, col="blue", xlab="driver's age", ylab="predictions",
↪  main="marginalized estimated frequencies")
lines(x=aggr$DrivAge, y=aggr$CANN, lwd=2, col="red")
legend(x="topright", pch=-1, lwd=2, col=c("blue","red"), legend=c("GLM", "CANN"))
```



- Marginally, for the driver's age covariate, the CANN does not seem to find new structure.
- Marginalized GLM improvements could also be achieved by a GAM that replaces linear functions by splines (on the link scale).

- Important: The above is only a marginal view, and it does not explicitly show interactions. The CANN also essentially improves on these interactions.
- This marginalized estimated frequencies are different from the partial dependence plot (PDP) – see notebook on explainability.
- Here, we empirically consider (under the population distribution $\boldsymbol{X} \sim \mathbb{P}$; for simplicity we drop the exposures v)

$$\mathbb{E}[\mu(\boldsymbol{X})|X_j = x_j] = \mathbb{E}[\mu(X_1, \ldots, X_q)|X_j = x_j],$$

  the PDP considers

$$\mathbb{E}[\mu(X_1, \ldots, X_{j-1}, x_j, X_{j+1}, \ldots, X_q)].$$

  The latter typically violates the true dependence structure within $\boldsymbol{X}$, whereas the former is a proper conditional expectation view under the population distribution $\mathbb{P}$.

Copyright

References I

Brauer, A. (2024) 'Enhancing actuarial non-life pricing models via transformers', *European Actuarial Journal*, 14(3), pp. 991–1012. Available at: https://doi.org/10.1007/s13385-024-00388-2.

Delong, Ł. and Wüthrich, M.V. (2025) 'Isotonic regression for variance estimation and its role in mean estimation and model validation', *North American Actuarial Journal*, 0(0), pp. 1–29. Available at: https://doi.org/10.1080/10920277.2024.2421221.

Havrylenko, Y. and Heger, J. (2024) 'Detection of interacting variables for generalized linear models via neural networks', *European Actuarial Journal*, 14(2), pp. 551–580. Available at: https://doi.org/10.1007/s13385-023-00362-4.

References II

He, K. *et al.* (2015) 'Deep residual learning for image recognition'. Available at: https://arxiv.org/abs/1512.03385.

Wüthrich, M.V. and Buser, C. (2016) 'Data analytics for non-life insurance pricing', *SSRN* [Preprint]. Available at: https://ssrn.com/abstract=2870308.

Wüthrich, M.V. and Merz, M. (2019) 'Editorial: Yes, we CANN!', *ASTIN Bulletin - The Journal of the IAA*, 49(1), pp. 1–3. Available at: https://www.cambridge.org/core/product/66E8BEC373B5CCEF3BF3303D442D6B75.

References III

Wüthrich, M.V. and Merz, M. (2023) *Statistical foundations of actuarial learning and its applications*. Springer. Available at: https://doi.org/10.1007/978-3-031-12409-9.